

# Tutorial on Probabilistic Techniques for Robot Navigation

Wolfram Burgard

---



# Some facts about the AIS Lab

- 6 senior researchers
- 32 Ph.D. students
- 2 project managers
- 1 technician, 1 secretary
- >300 publications
- 4 running EC funded projects
- Head of the DFG Cluster of Excellence BrainLinks-BrainTools



# Fields of Research

- Mobile robotics
- State estimation and modeling
- Mapping
- Decision-theoretic approaches
- Adaptive techniques and learning
- Scene understanding
- Mobile manipulation
- Multi-robot coordination
- Robots and embedded systems
- Autonomous cars
- Flying vehicles
- ...
- **Probabilistic robotics**



# Probabilistic Techniques in Robotics

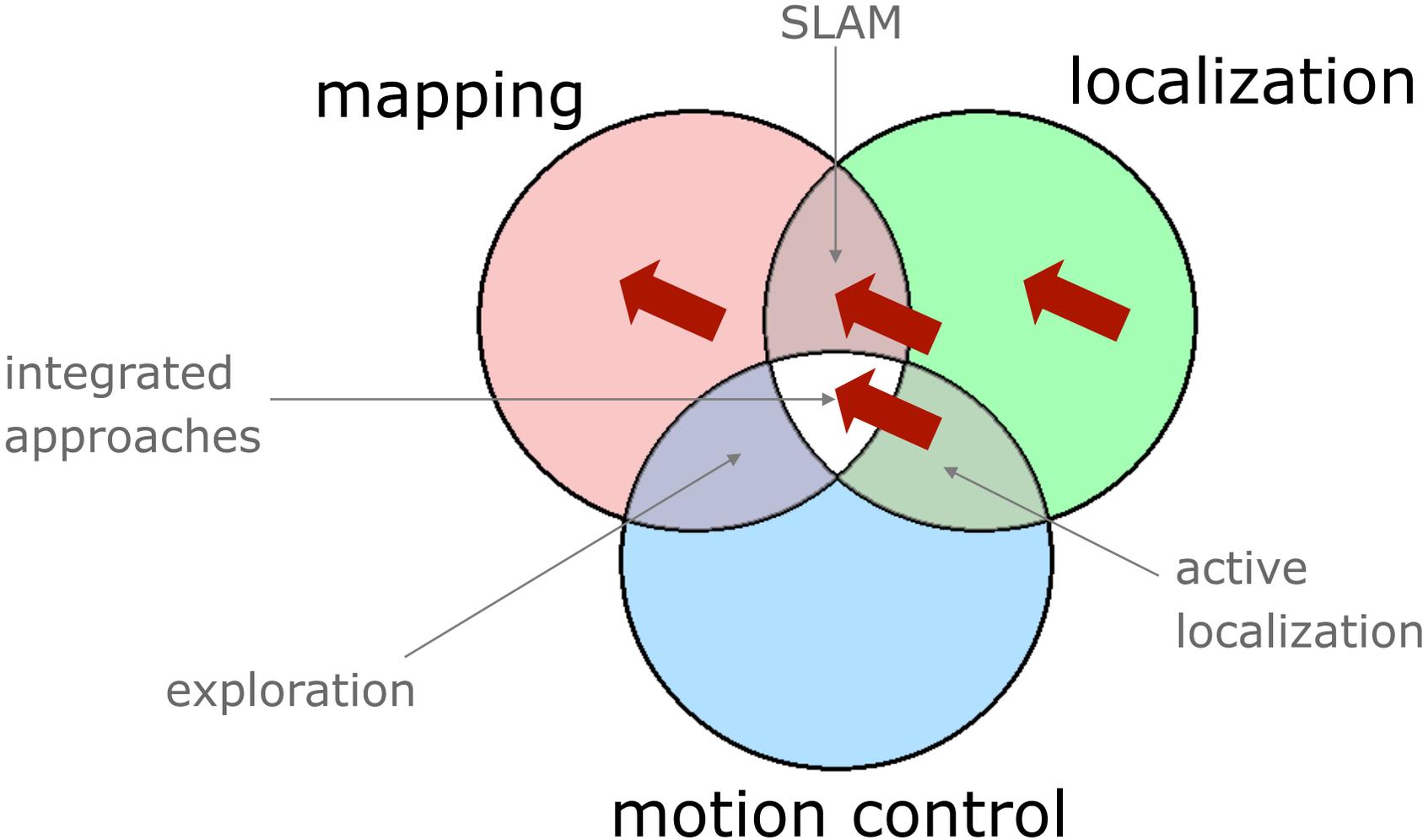
- Perception = state estimation
- Action = utility maximization

## Key Questions

- Representation
- Maximization

especially in the context of higher dimensions

# Dimensions of Mobile Robot Navigation



# Topics in this Talk

- State estimation
- Localization
- Mapping
- SLAM
- Exploration
  
- Applications

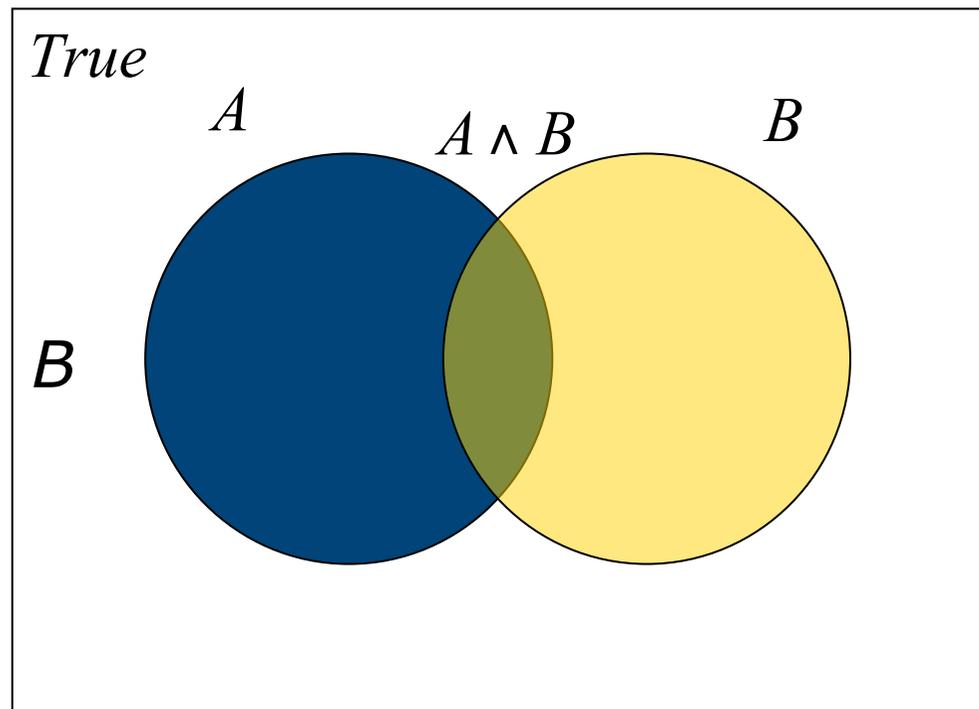
# Axioms of Probability Theory

$P(A)$  denotes probability that proposition  $A$  is true.

- $0 \leq P(A) \leq 1$
- $P(\textit{True}) = 1$                        $P(\textit{False}) = 0$
- $P(A \vee B) = P(A) + P(B) - P(A \wedge B)$

## A Closer Look at Axiom 3

$$P(A \vee B) = P(A) + P(B) - P(A \wedge B)$$



# Bayes Formula

$$P(x, y) = P(x | y)P(y) = P(y | x)P(x)$$

$\Rightarrow$

$$P(x | y) = \frac{P(y | x) P(x)}{P(y)} = \frac{\text{likelihood} \cdot \text{prior}}{\text{evidence}}$$

# Recursive Bayesian Updating

$$P(x | z_1, \dots, z_n) = \frac{P(z_n | x, z_1, \dots, z_{n-1}) P(x | z_1, \dots, z_{n-1})}{P(z_n | z_1, \dots, z_{n-1})}$$

**Markov assumption:**

$z_n$  is independent of  $z_1, \dots, z_{n-1}$  given  $x$

$$\begin{aligned} P(x | z_1, \dots, z_n) &= \frac{P(z_n | x) P(x | z_1, \dots, z_{n-1})}{P(z_n | z_1, \dots, z_{n-1})} \\ &= \eta P(z_n | x) P(x | z_1, \dots, z_{n-1}) \\ &= \eta_{1\dots n} \left[ \prod_{i=1}^n P(z_i | x) \right] P(x) \end{aligned}$$

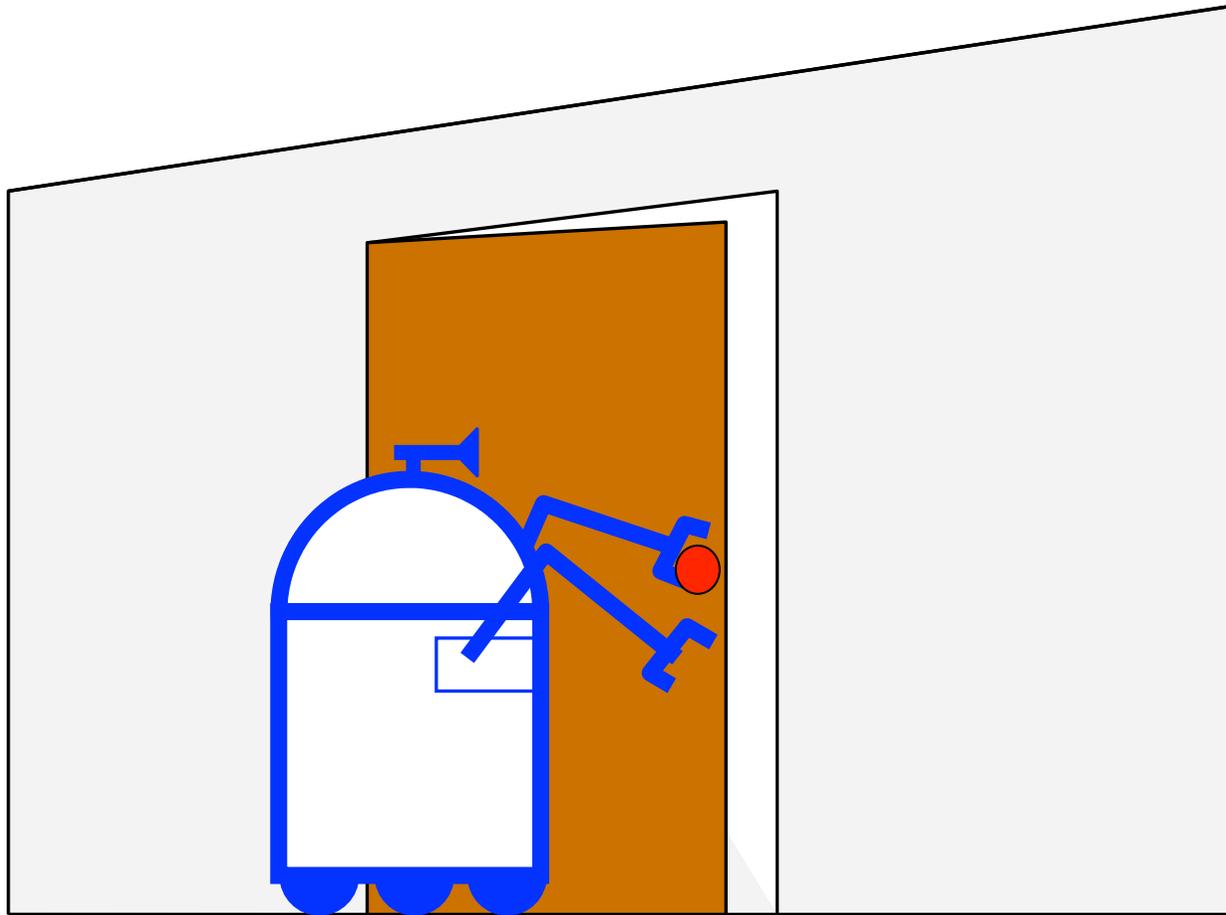
## Modeling Actions

- To incorporate the outcome of an action  $u$  into the current “belief”, we use the conditional pdf

$$P(x|u,x')$$

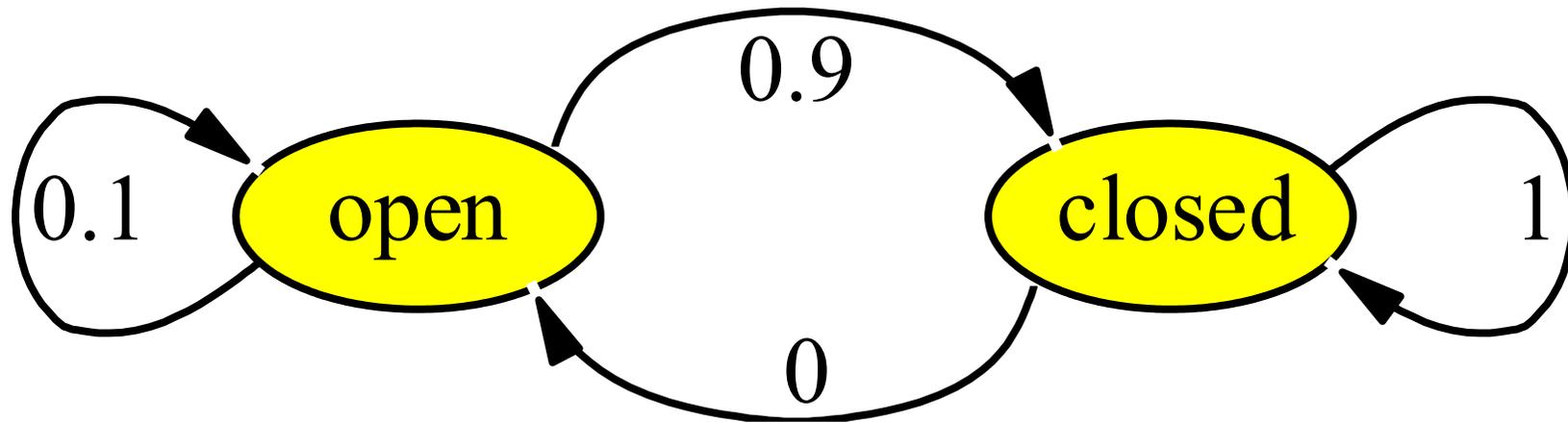
- This term specifies the pdf that **executing  $u$  changes the state from  $x'$  to  $x$ .**

# Example: Closing the door



## State Transitions

$P(x|u, x')$  for  $u =$  “close door”:



If the door is open, the action “close door” succeeds in 90% of all cases

# Integrating the Outcome of Actions

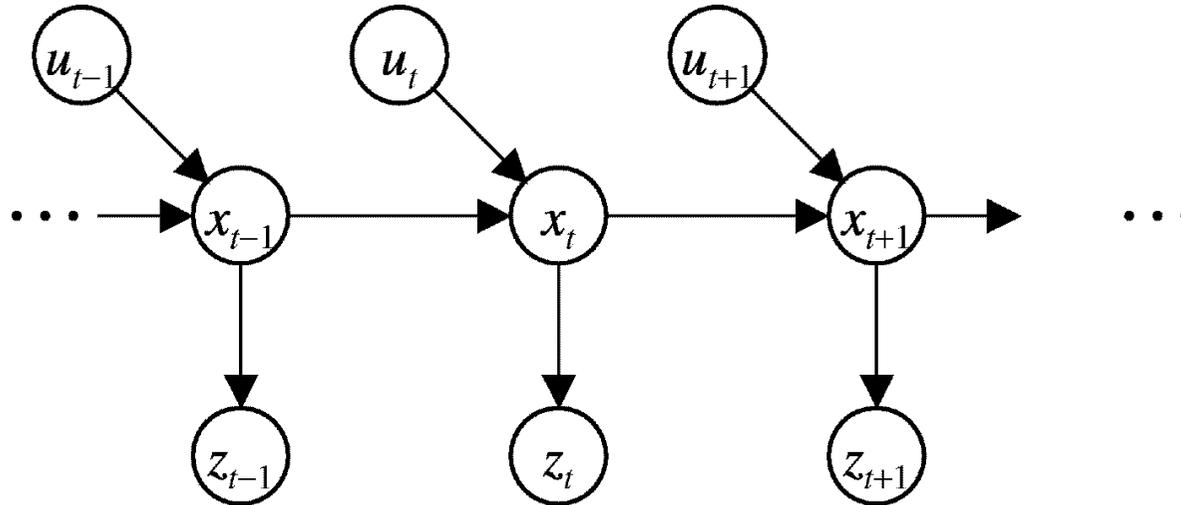
Continuous case:

$$P(x | u) = \int P(x | u, x')P(x') dx'$$

Discrete case:

$$P(x | u) = \sum P(x | u, x')P(x')$$

# Markov Assumption



$$p(z_t \mid x_{0:t}, z_{1:t-1}, u_{1:t}) = p(z_t \mid x_t)$$

$$p(x_t \mid x_{0:t-1}, z_{1:t-1}, u_{1:t}) = p(x_t \mid x_{t-1}, u_t)$$

## Underlying Assumptions

- Static world
- Independent noise
- Perfect model, no approximation errors

# Bayes Filters

$z$  = observation  
 $u$  = action  
 $x$  = state

$$\boxed{Bel(x_t)} = P(x_t | u_1, z_1, \dots, u_t, z_t)$$

**Bayes**  $= \eta P(z_t | x_t, u_1, z_1, \dots, u_t) P(x_t | u_1, z_1, \dots, u_t)$

**Markov**  $= \eta P(z_t | x_t) P(x_t | u_1, z_1, \dots, u_t)$

**Total prob.**  $= \eta P(z_t | x_t) \int P(x_t | u_1, z_1, \dots, u_t, x_{t-1})$   
 $P(x_{t-1} | u_1, z_1, \dots, u_t) dx_{t-1}$

**Markov**  $= \eta P(z_t | x_t) \int P(x_t | u_t, x_{t-1}) P(x_{t-1} | u_1, z_1, \dots, u_t) dx_{t-1}$

**Markov**  $= \eta P(z_t | x_t) \int P(x_t | u_t, x_{t-1}) P(x_{t-1} | u_1, z_1, \dots, z_{t-1}) dx_{t-1}$

$$\boxed{= \eta P(z_t | x_t) \int P(x_t | u_t, x_{t-1}) Bel(x_{t-1}) dx_{t-1}}$$

$$Bel(x_t) = \eta P(z_t | x_t) \int P(x_t | u_t, x_{t-1}) Bel(x_{t-1}) dx_{t-1}$$

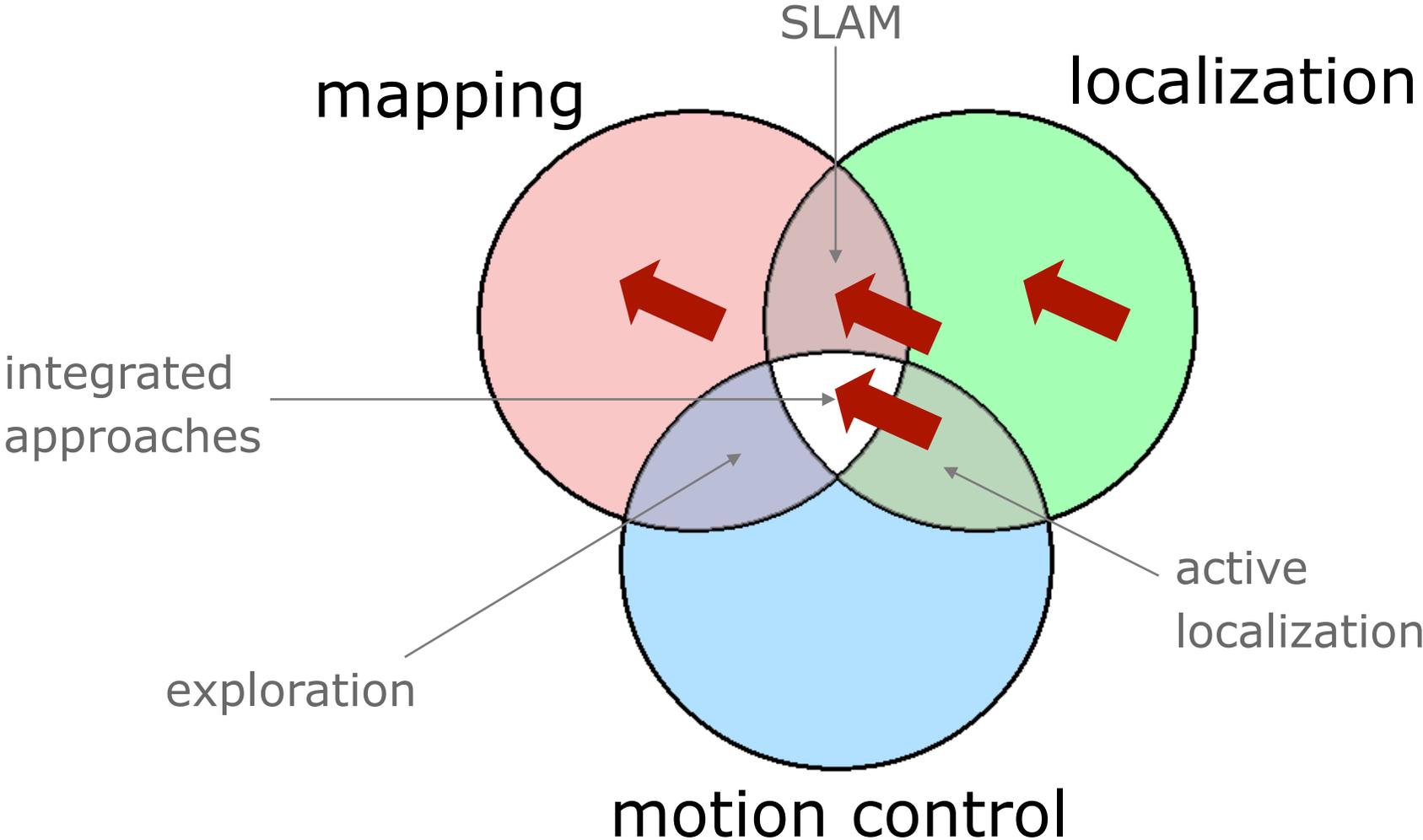
1. Algorithm **Bayes\_filter**(  $Bel(x), d$  ):
2.  $\eta = 0$
3. If  $d$  is a perceptual data item  $z$  then
4.     For all  $x$  do
5.          $Bel'(x) = P(z | x) Bel(x)$
6.          $\eta = \eta + Bel'(x)$
7.     For all  $x$  do
8.          $Bel'(x) = \eta^{-1} Bel'(x)$
9. Else if  $d$  is an action data item  $u$  then
10.     For all  $x$  do
11.          $Bel'(x) = \int P(x | u, x') Bel(x') dx'$
12. Return  $Bel'(x)$

# Bayes Filters are Familiar!

$$Bel(x_t) = \eta p(z_t | x_t) \int p(x_t | u_t, x_{t-1}) Bel(x_{t-1}) dx_{t-1}$$

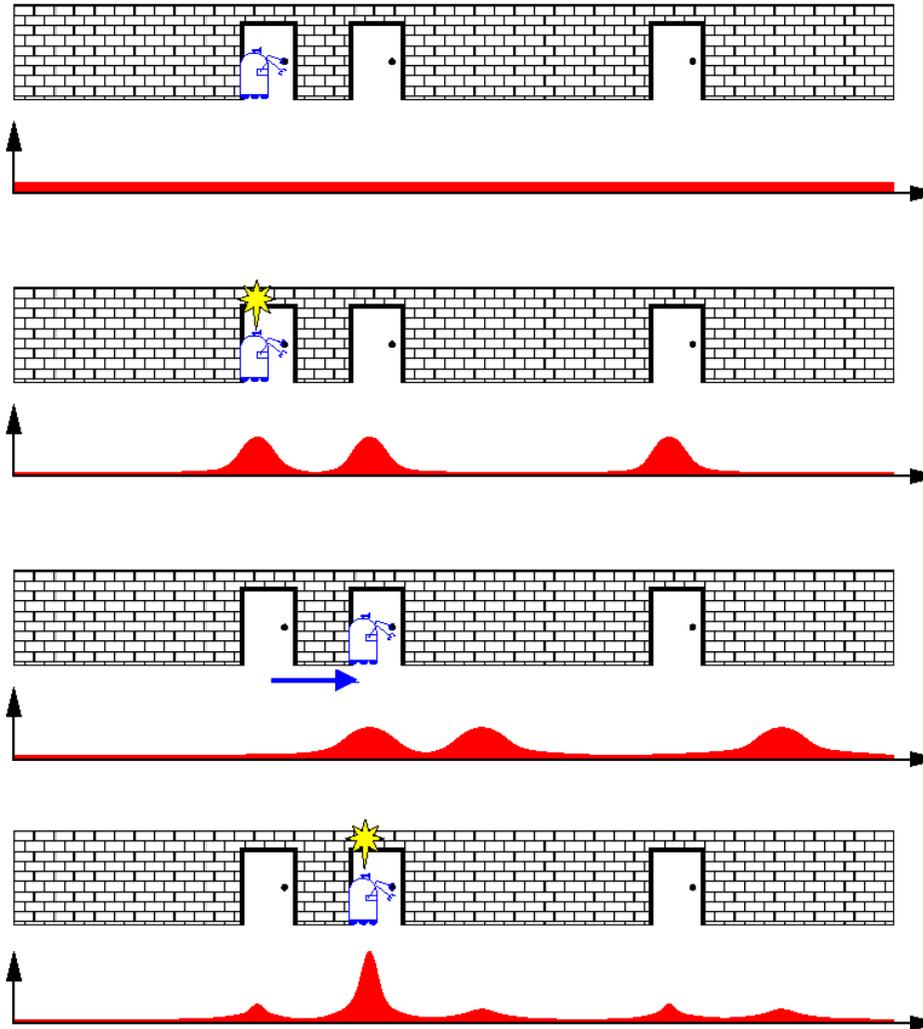
- Kalman filters
- Particle filters
- Hidden Markov models
- Dynamic Bayesian networks
- Partially Observable Markov Decision Processes (POMDPs)

# Dimensions of Mobile Robot Navigation



# Probabilistic Localization

$$Bel(x | z, u) = \alpha p(z | x) \int_{x'} p(x | u, x') Bel(x') dx'$$



# There are Different Representations

- Kalman filters
- Multi-hypothesis tracking
- Grid-based representations
- Topological approaches
- Particle filters

# Idea of a Particle Filter

Represent the posterior by a set of weighted samples.

$$S = \left\{ \left\langle s^{[i]}, w^{[i]} \right\rangle \mid i = 1, \dots, N \right\}$$

State hypothesis                      Importance weight

- Particles are propagated according to the motion model  $p(x \mid u, x')$
- They are weighted according to the observation likelihood  $p(z \mid x)$
- They survive with a probability proportional to their importance weight  $w$

# Representations: Particle Filter

$$Bel(x | z, u) = \alpha p(z | x) \int_{x'} p(x | u, x') Bel(x') dx'$$

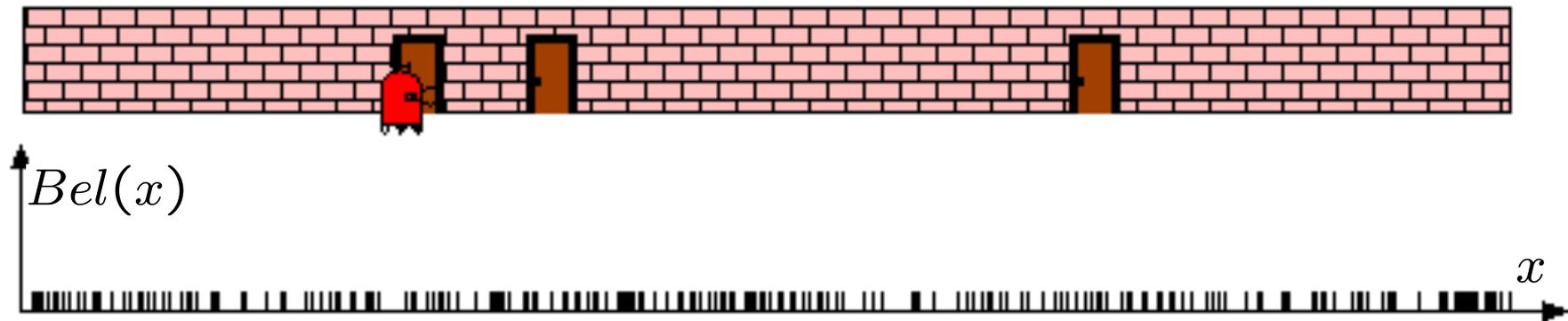
1. Draw  $x'$  from  $Bel(x)$

2. Draw  $x$  from  $p(x | u, x')$

3. Importance factor for  $x$  :  $w = \alpha p(z | x)$

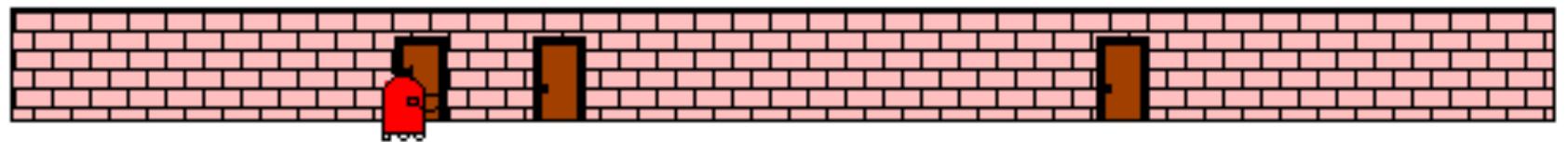
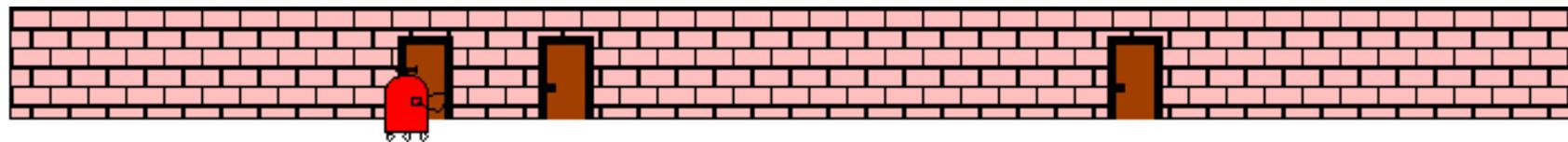
4. Re-sample

# Mobile Robot Localization with Particle Filters



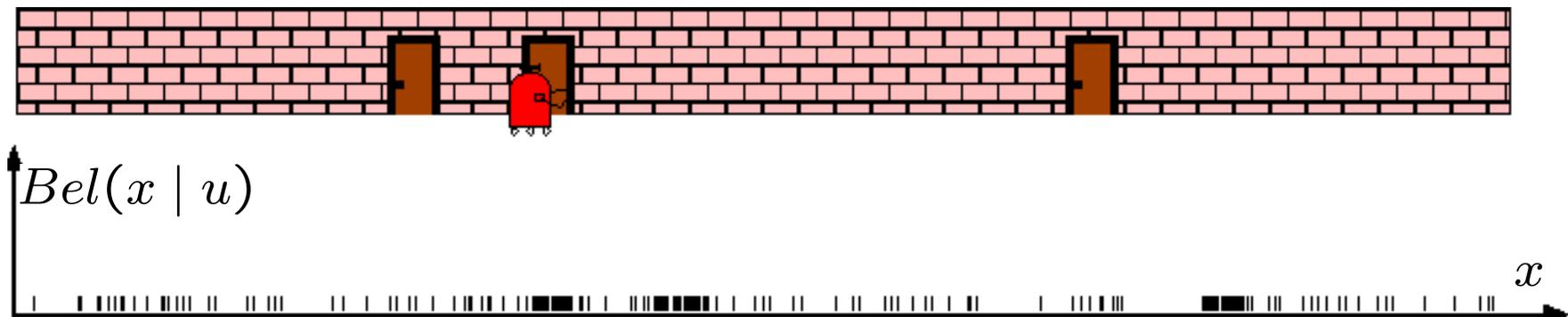
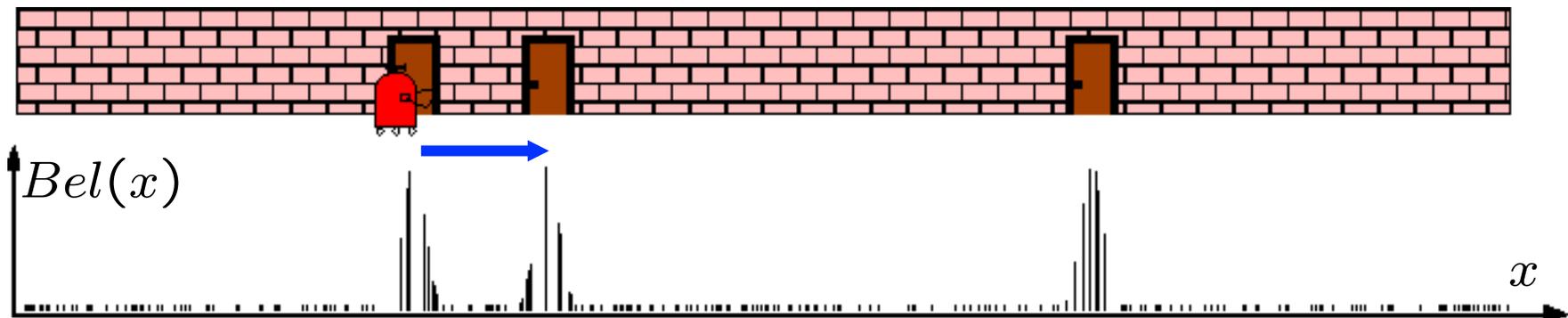
# Sensor Update

$$Bel(x | z) = \alpha p(z | x) Bel(x)$$



# Robot Motion & Resampling

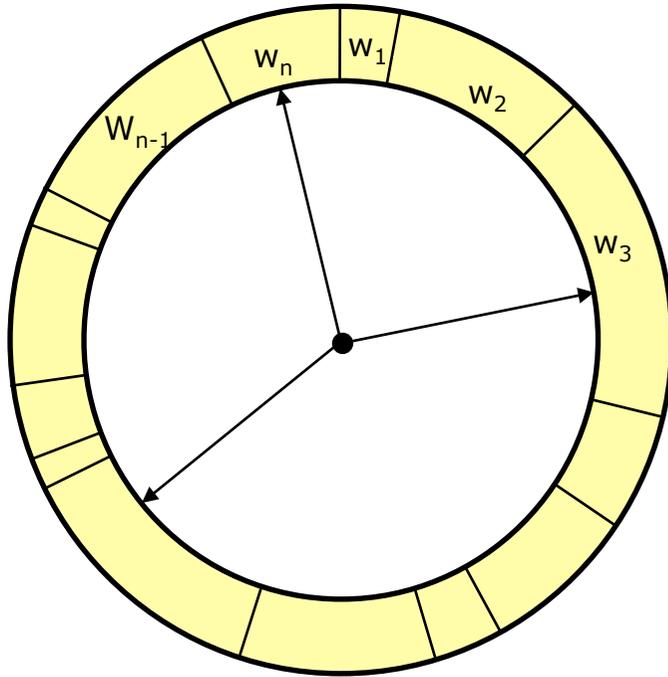
$$Bel(x | u) = \int_{x'} p(x | u, x') Bel(x')$$



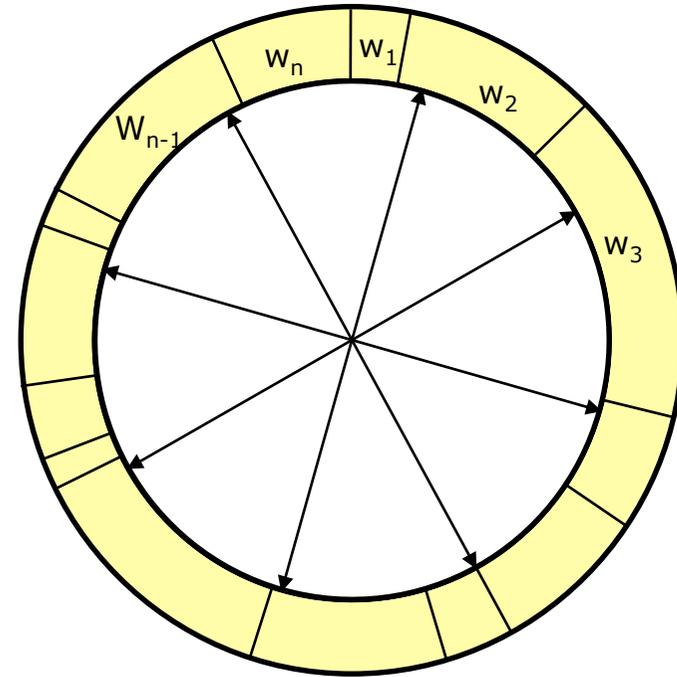
# Particle Filter Algorithm

1. Algorithm **particle\_filter**(  $S_{t-1}, u_{t-1}, z_t$ ):
2.  $S_t = \emptyset, \eta = 0$
3. **For**  $i = 1 \dots n$  *Generate new samples*
4. Sample index  $j(i)$  from the discrete distribution given by  $w_{t-1}$
5. Sample  $x_t^i$  from  $p(x_t | x_{t-1}, u_{t-1})$  using  $x_{t-1}^{j(i)}$  and  $u_{t-1}$
6.  $w_t^i = p(z_t | x_t^i)$  *Compute importance weight*
7.  $\eta = \eta + w_t^i$  *Update normalization factor*
8.  $S_t = S_t \cup \{ \langle x_t^i, w_t^i \rangle \}$  *Insert*
9. **For**  $i = 1 \dots n$
10.  $w_t^i = w_t^i / \eta$  *Normalize weights*

# Resampling

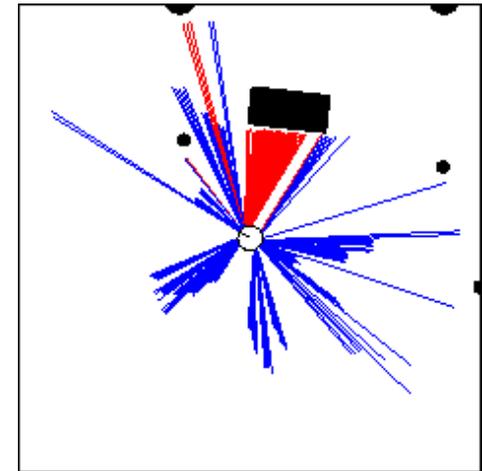
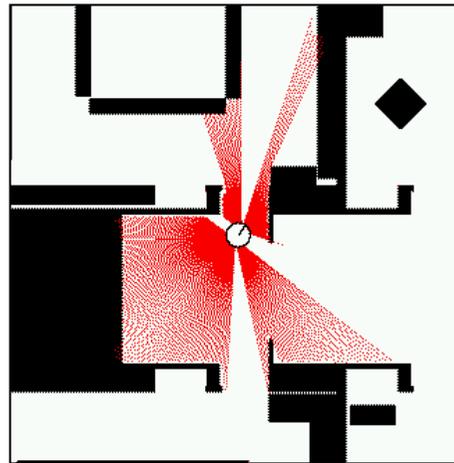
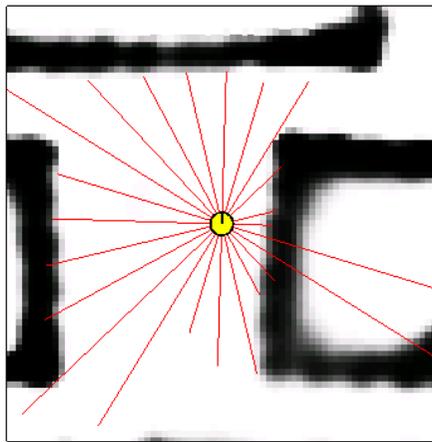


- Roulette wheel
- Binary search,  $n \log n$



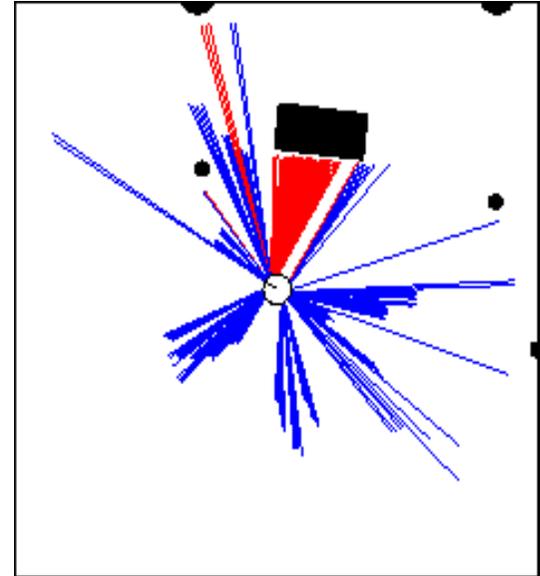
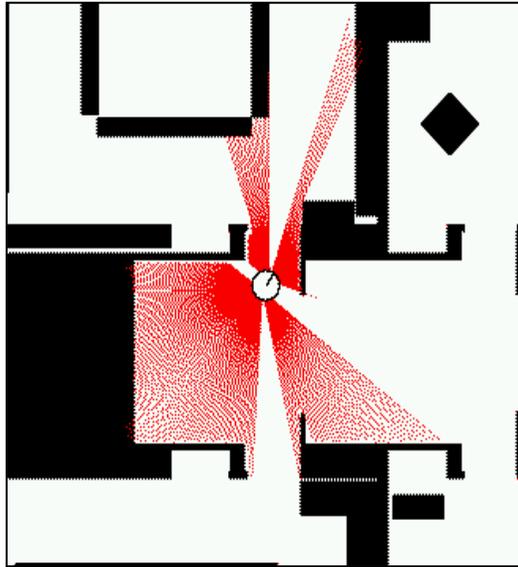
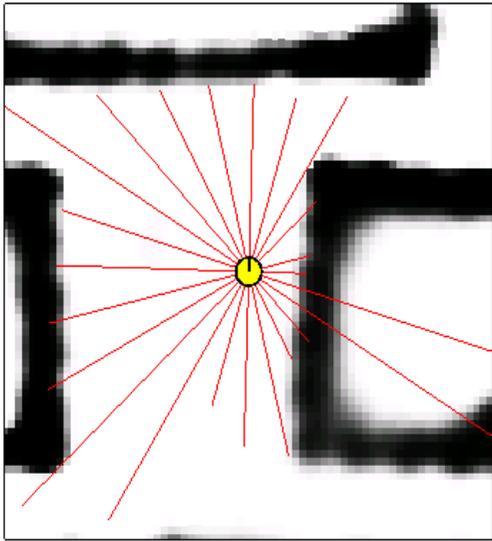
- Stochastic universal sampling
- Systematic resampling
- Linear time complexity
- Easy to implement, low variance

# Sensor Models for Proximity Sensors



- The central task is to determine  $P(z|x)$ , i.e., the probability of a measurement  $z$  given that the robot is at position  $x$ .
- **Question:** Where do the probabilities come from?
- **Approach:** Let's try to explain a measurement.

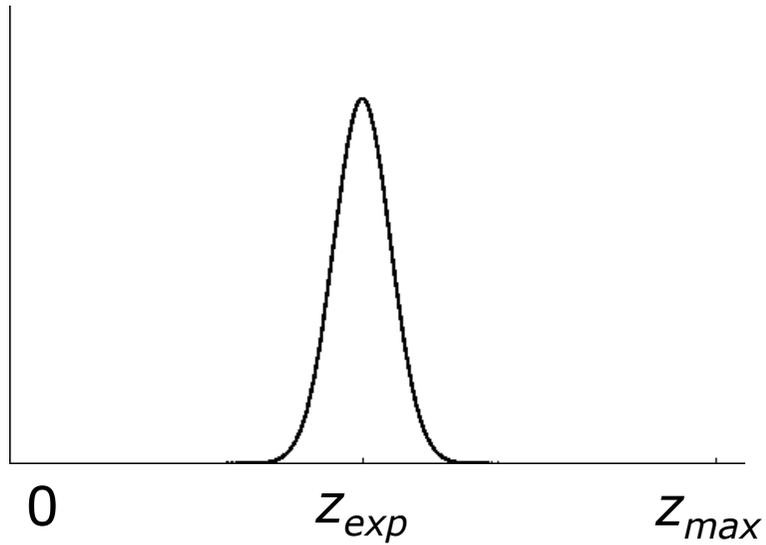
# Conditional Independence



$$P(z | x, m) = \prod_{k=1}^K P(z_k | x, m)$$

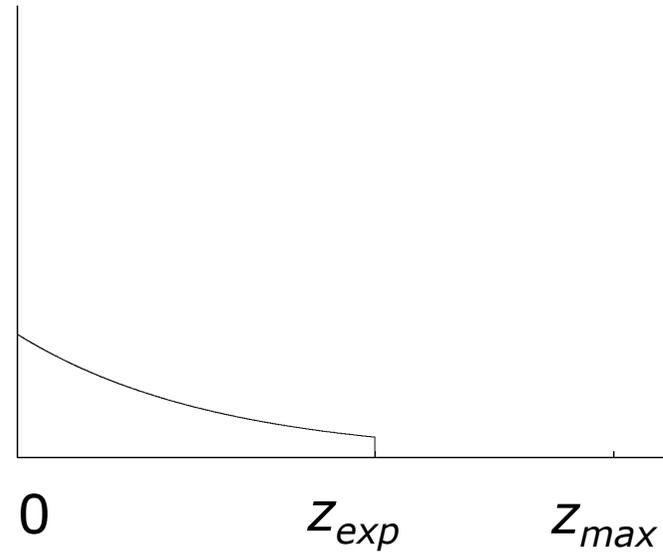
# Beam-based Proximity Model

Measurement noise



$$P_{hit}(z | x, m) = \eta \frac{1}{\sqrt{2\pi b}} e^{-\frac{1}{2} \frac{(z - z_{exp})^2}{b}}$$

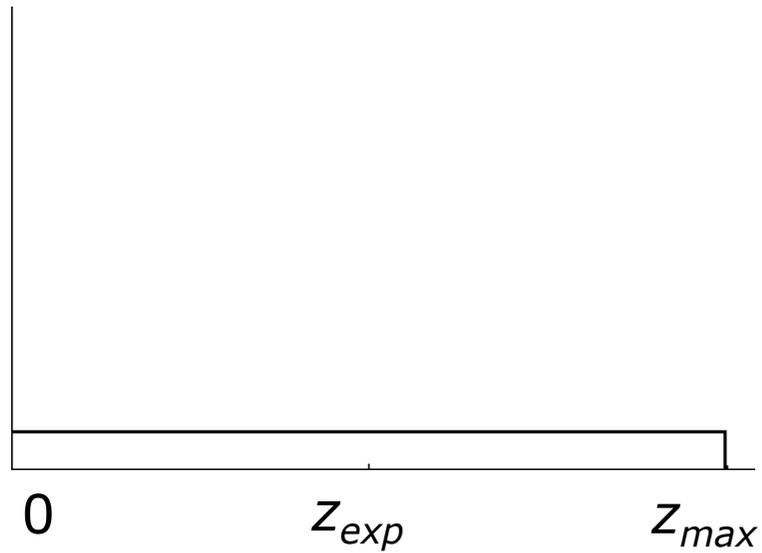
Unexpected obstacles



$$P_{unexp}(z | x, m) = \left\{ \begin{array}{ll} \eta \lambda e^{-\lambda z} & z < z_{exp} \\ 0 & otherwise \end{array} \right\}$$

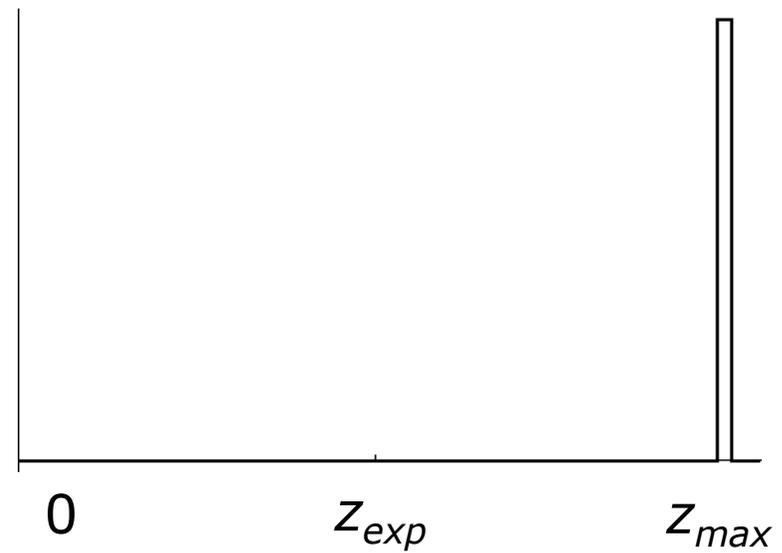
# Beam-based Proximity Model

Random measurement



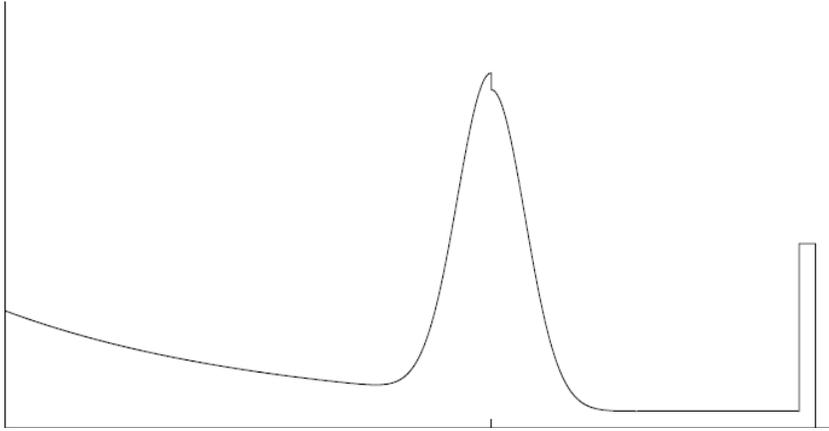
$$P_{rand}(z | x, m) = \eta \frac{1}{z_{max}}$$

Max range



$$P_{max}(z | x, m) = \eta \frac{1}{z_{small}}$$

# Resulting Mixture Density



$$P(z | x, m) = \begin{pmatrix} \alpha_{\text{hit}} \\ \alpha_{\text{unexp}} \\ \alpha_{\text{max}} \\ \alpha_{\text{rand}} \end{pmatrix}^T \cdot \begin{pmatrix} P_{\text{hit}}(z | x, m) \\ P_{\text{unexp}}(z | x, m) \\ P_{\text{max}}(z | x, m) \\ P_{\text{rand}}(z | x, m) \end{pmatrix}$$

How can we determine the model parameters?

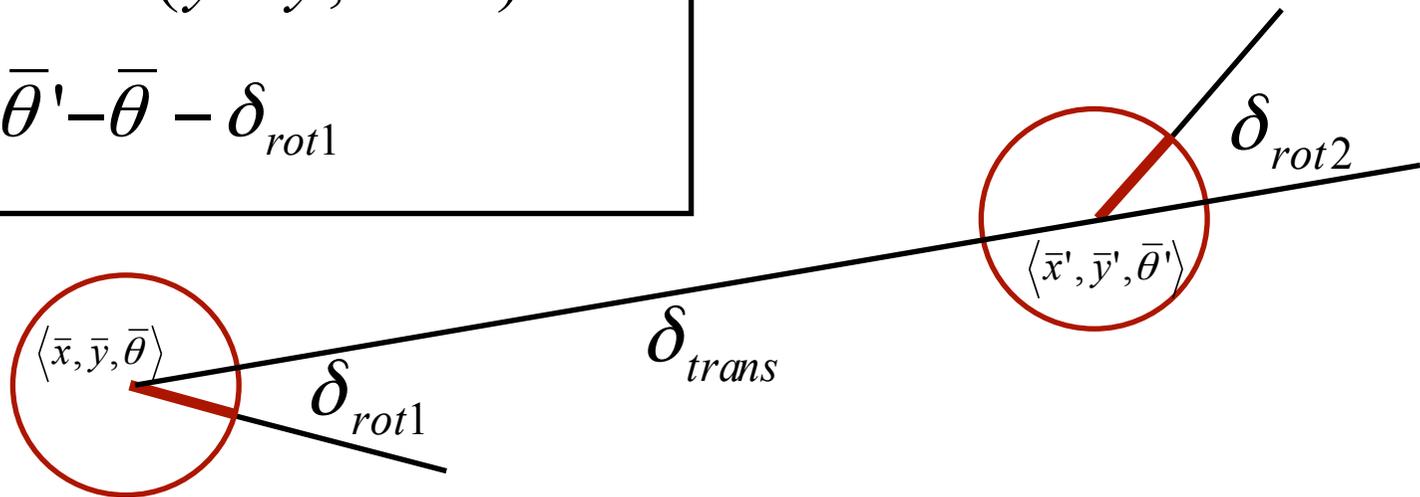
# Odometry Motion Model

- Robot moves from  $\langle \bar{x}, \bar{y}, \bar{\theta} \rangle$  to  $\langle \bar{x}', \bar{y}', \bar{\theta}' \rangle$ .
- Odometry information  $u = \langle \delta_{rot1}, \delta_{rot2}, \delta_{trans} \rangle$ .

$$\delta_{trans} = \sqrt{(\bar{x}' - \bar{x})^2 + (\bar{y}' - \bar{y})^2}$$

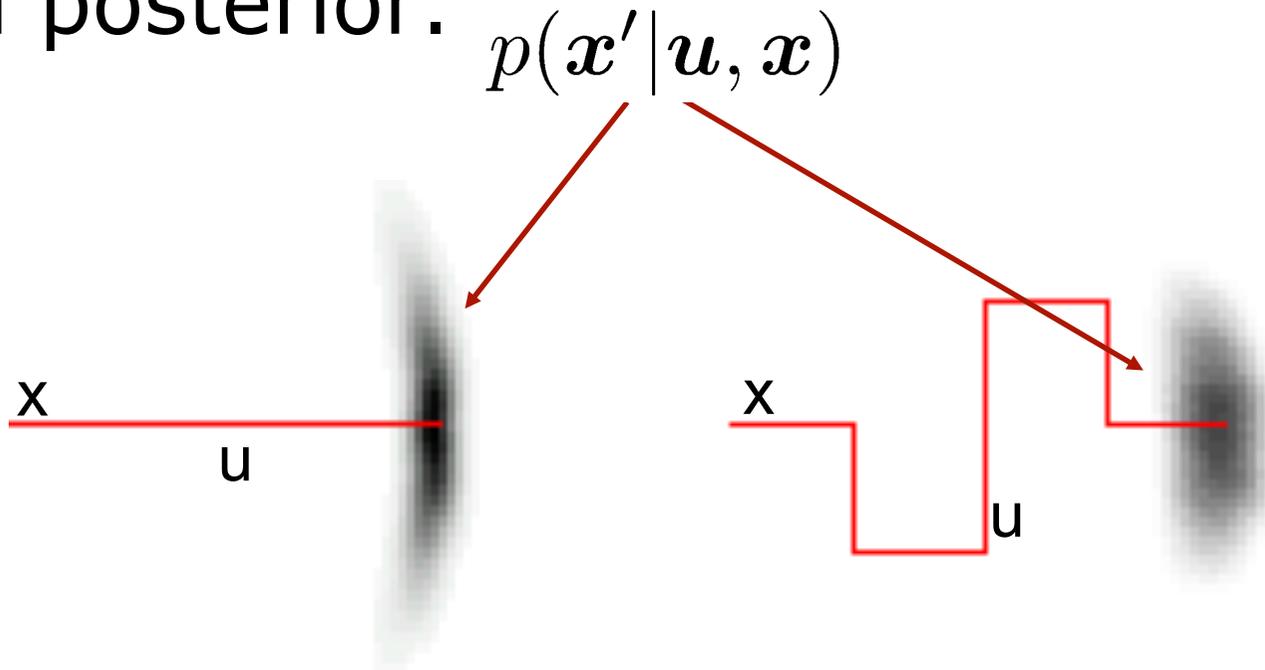
$$\delta_{rot1} = \text{atan2}(\bar{y}' - \bar{y}, \bar{x}' - \bar{x}) - \bar{\theta}$$

$$\delta_{rot2} = \bar{\theta}' - \bar{\theta} - \delta_{rot1}$$

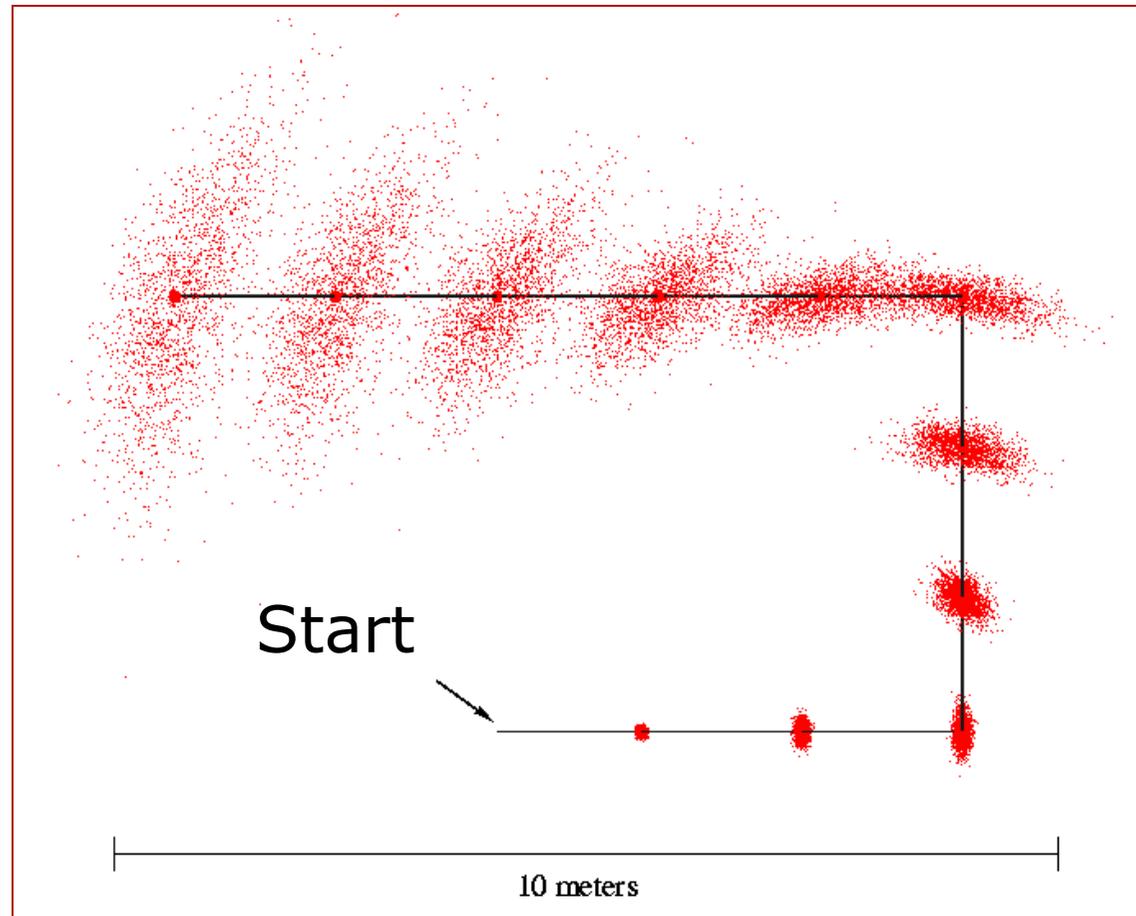


# Application

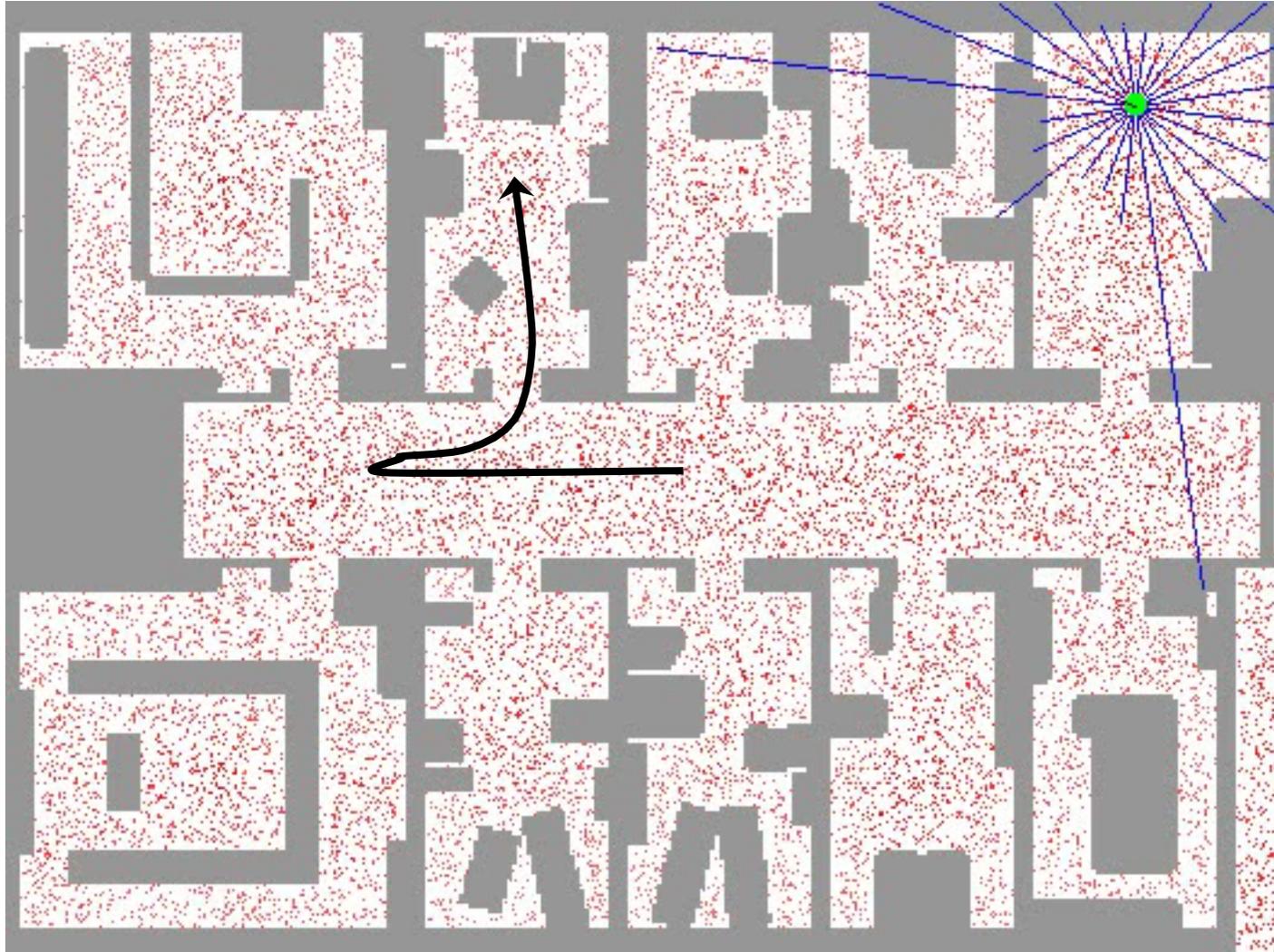
- Repeated application of the motion model for short movements.
- Typical banana-shaped distributions obtained for the 2d-projection of the 3d posterior.



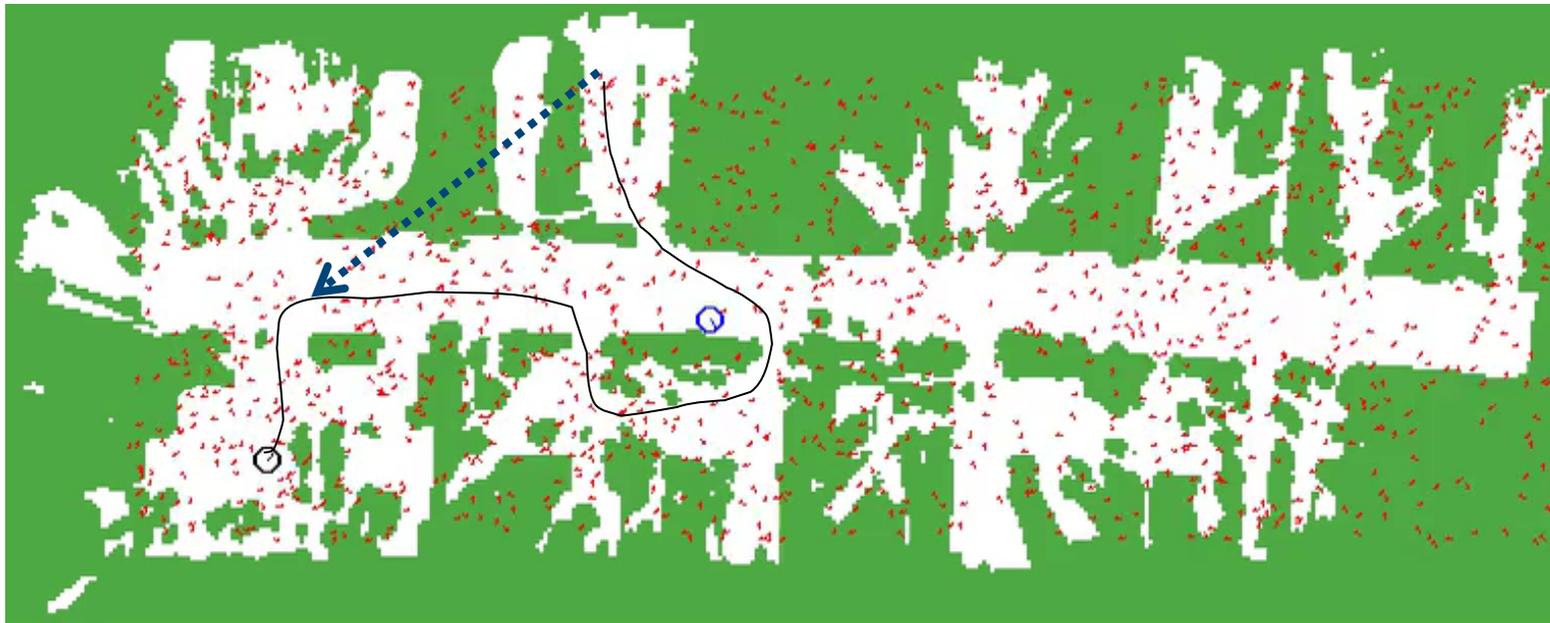
# Sampling from Our Motion Model



# MCL: Global Localization (Sonar)



# Vision-based Localization



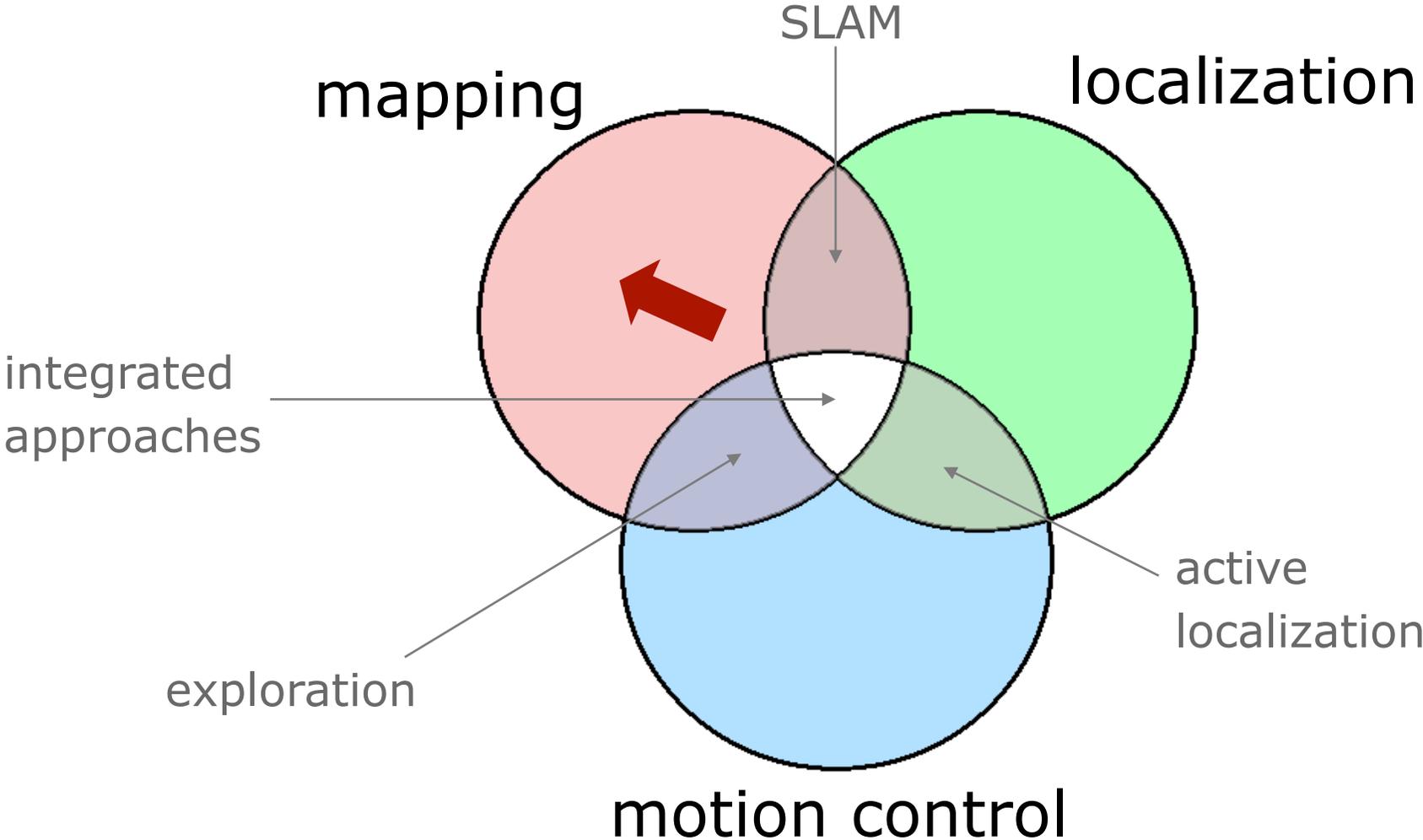
# Why Navigation is Relevant: Self-Driving Cars



# Precise Localization and Positioning for Mobile Robots



# Dimensions of Mobile Robot Navigation



# Occupancy Grid Maps

- Introduced by Moravec and Elfes in 1985
- Represent environment by a grid.
- Estimate the probability that a location is occupied by an obstacle.
- **Key assumptions**
  - Occupancy of individual cells  $m_t^{[xy]}$  is independent

$$\begin{aligned} Bel(m_t) &= P(m_t \mid u_1, z_2 \dots, u_{t-1}, z_t) \\ &= \prod_{x,y} Bel(m_t^{[xy]}) \end{aligned}$$

- Robot positions are known!

# Updating Occupancy Grid Maps

- Update the map cells using the **inverse sensor model**

$$Bel(m_t^{[xy]}) = 1 - \left( 1 + \frac{P(m_t^{[xy]} | z_t, u_{t-1})}{1 - P(m_t^{[xy]} | z_t, u_{t-1})} \cdot \frac{1 - P(m_t^{[xy]})}{P(m_t^{[xy]})} \cdot \frac{Bel(m_{t-1}^{[xy]})}{1 - Bel(m_{t-1}^{[xy]})} \right)^{-1}$$

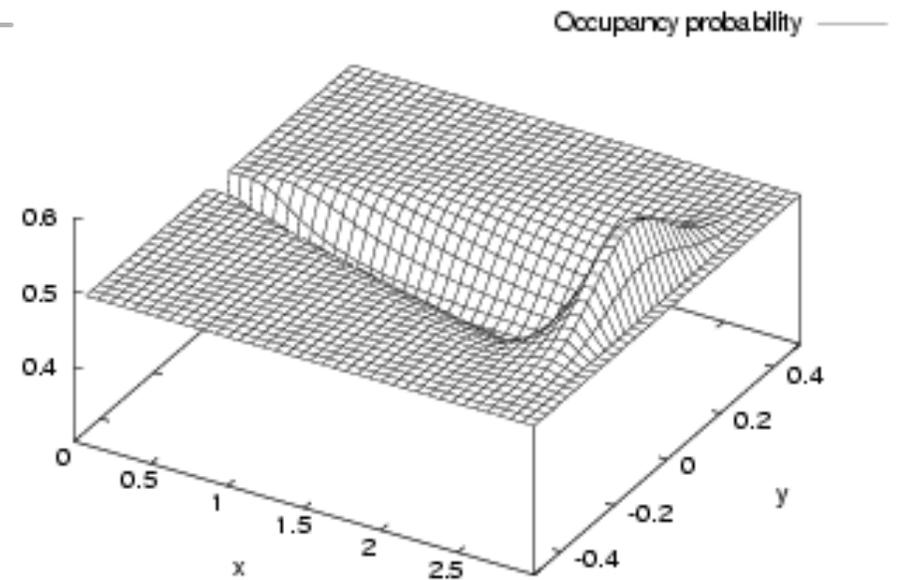
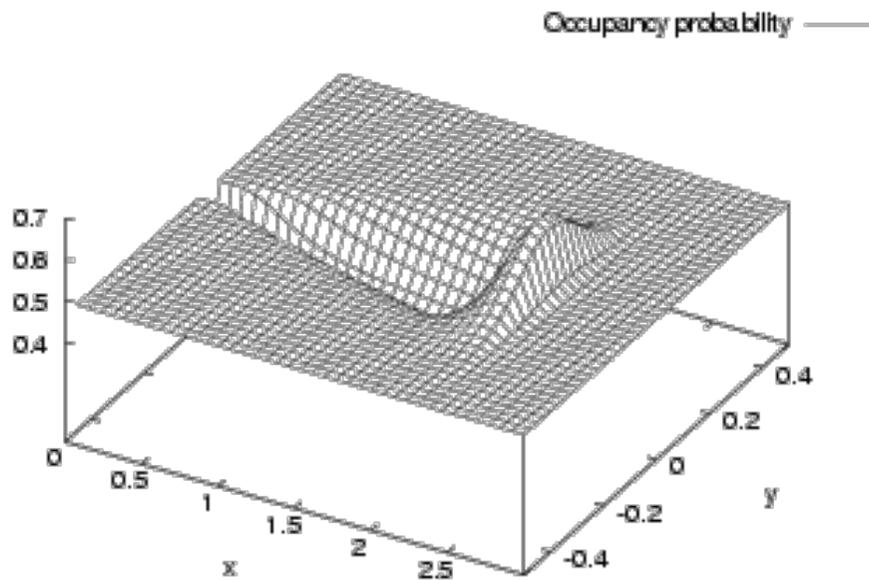
- Or use the **log-odds representation**

$$\bar{B}(m_t^{[xy]}) = \log odds(m_t^{[xy]} | z_t, u_{t-1}) - \log odds(m_t^{[xy]}) + \bar{B}(m_{t-1}^{[xy]})$$

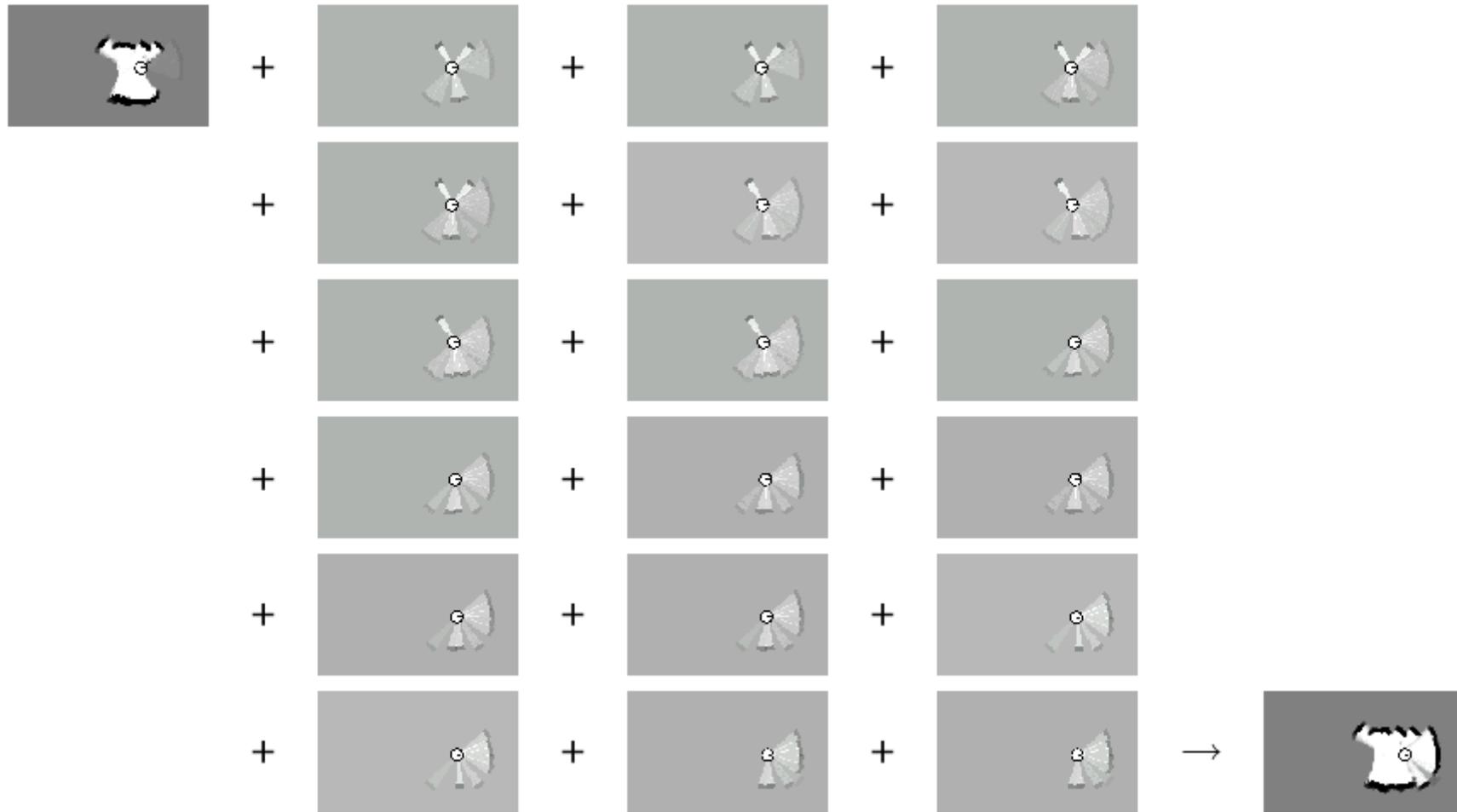
$$\bar{B}(m_0^{[xy]}) := \log odds(m_0^{[xy]}) \quad odds(x) := \left( \frac{P(x)}{1 - P(x)} \right)$$

# Typical Sensor Model for Ultrasound Sensors

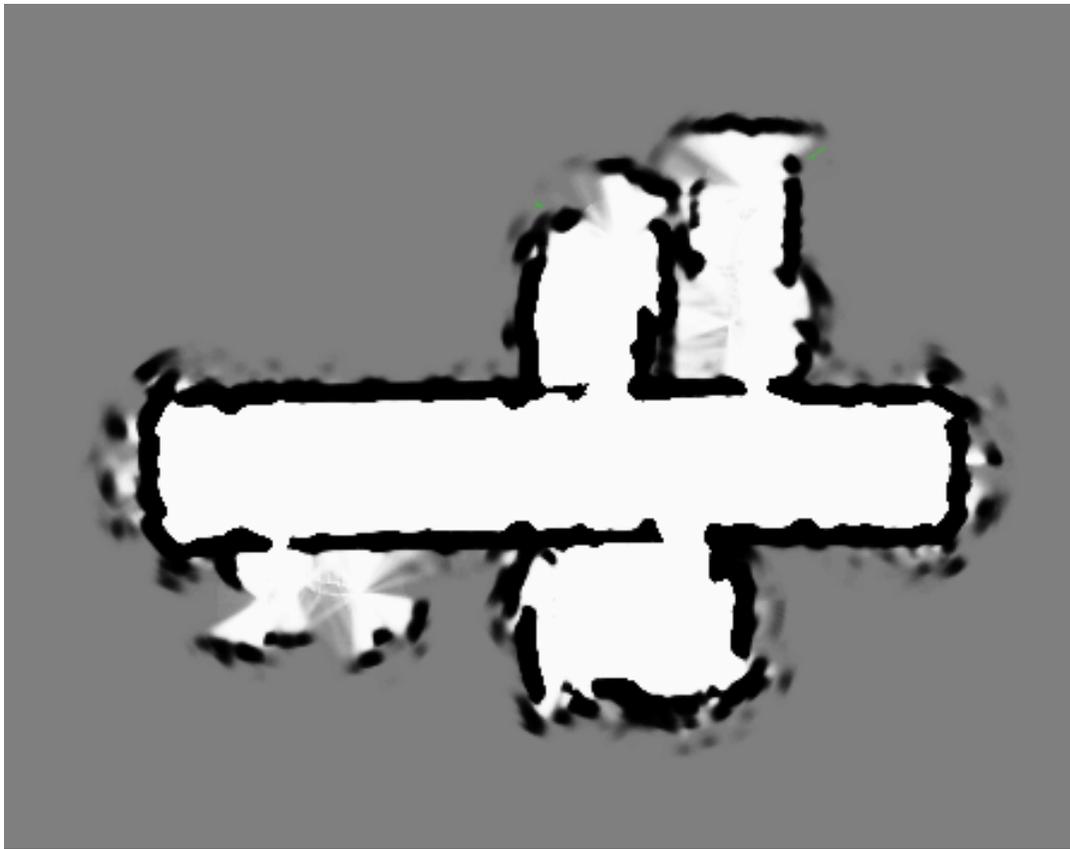
$P(m_t^{[xy]} | z_t, u_{t-1})$  as a mixture of a linear function  
and a Gaussian:



# Incremental Updating of Occupancy Grids (Example)



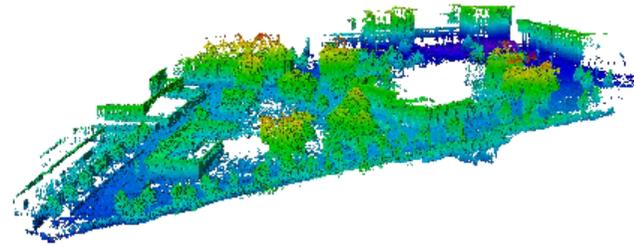
# Resulting Map Obtained with Ultrasound Sensors



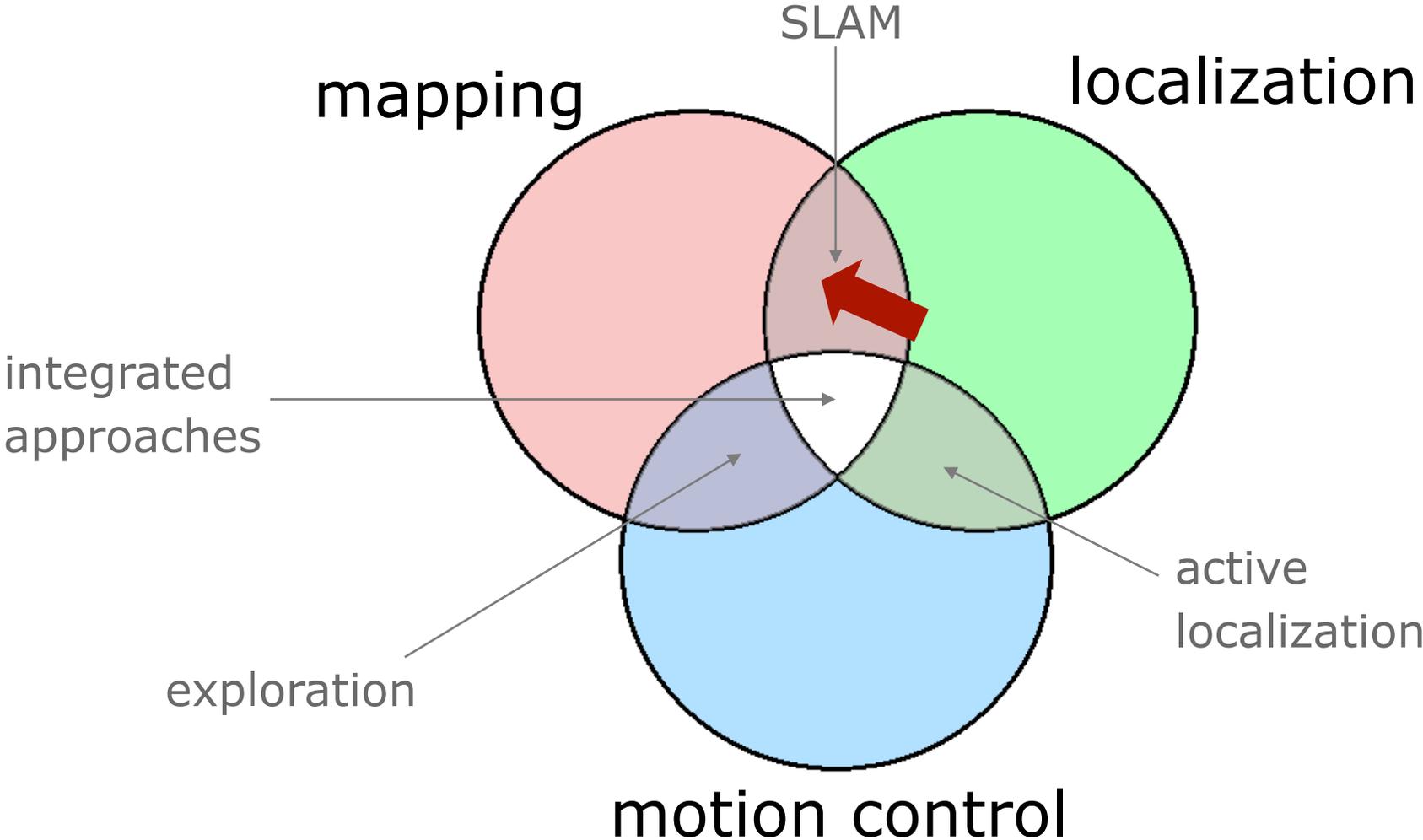
# 3D Occupancy Map (OctoMap)

Freiburg computer science campus

(292 x 167 x 28 m<sup>3</sup>, 20 cm resolution)



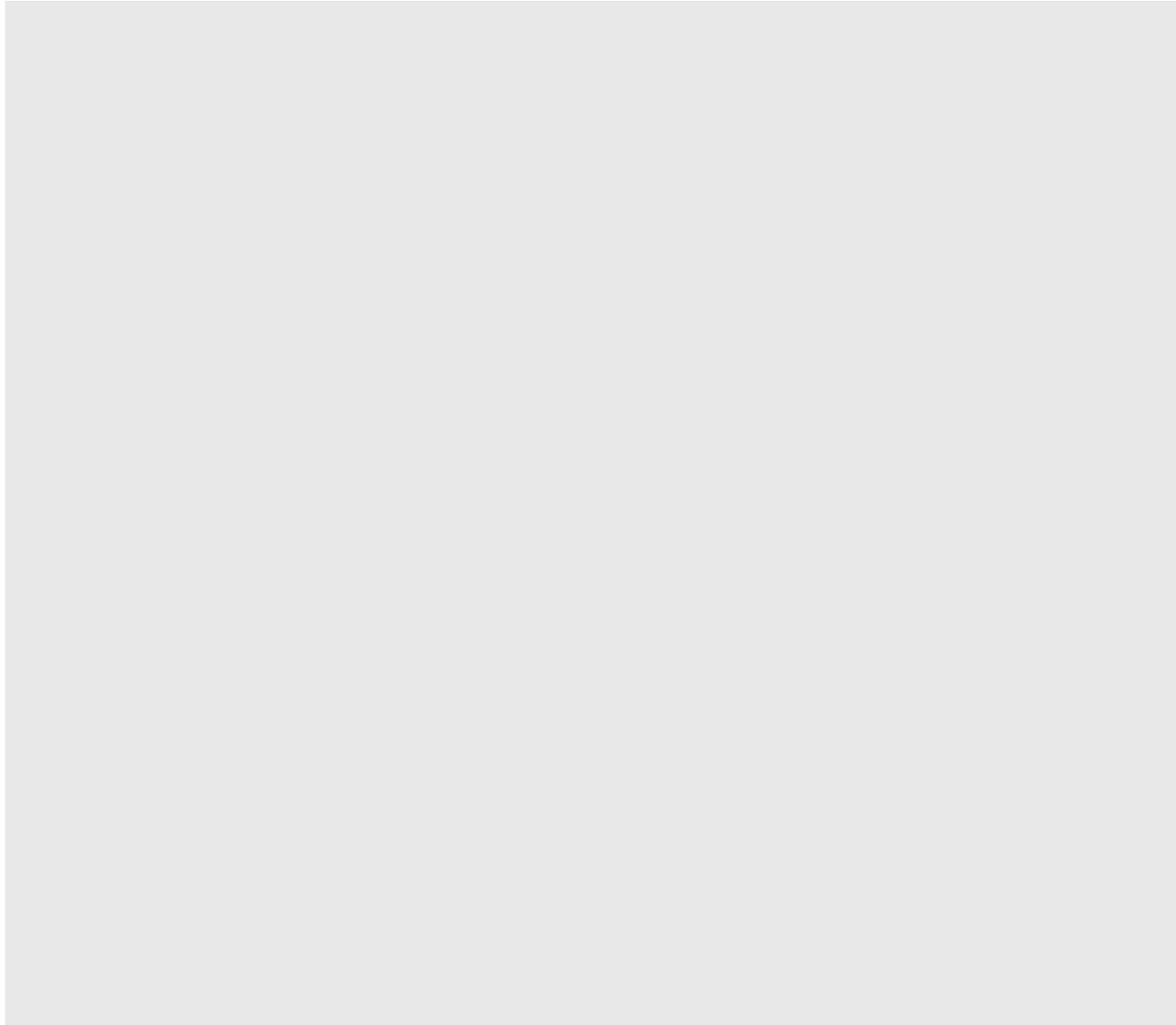
# Dimensions of Mobile Robot Navigation



# Simultaneous Localization and Mapping (SLAM)

- To determine its position, the **robot needs a map**.
- During mapping, the robot **needs to know its position** to learn a consistent model
- Simultaneous localization and mapping (SLAM) is a “**chicken and egg problem**”

# Why SLAM is Hard: Raw Odometry



# Scan Matching

Maximize the likelihood of the  $t$ -th pose and map relative to the pose and the map at  $t-1$ .

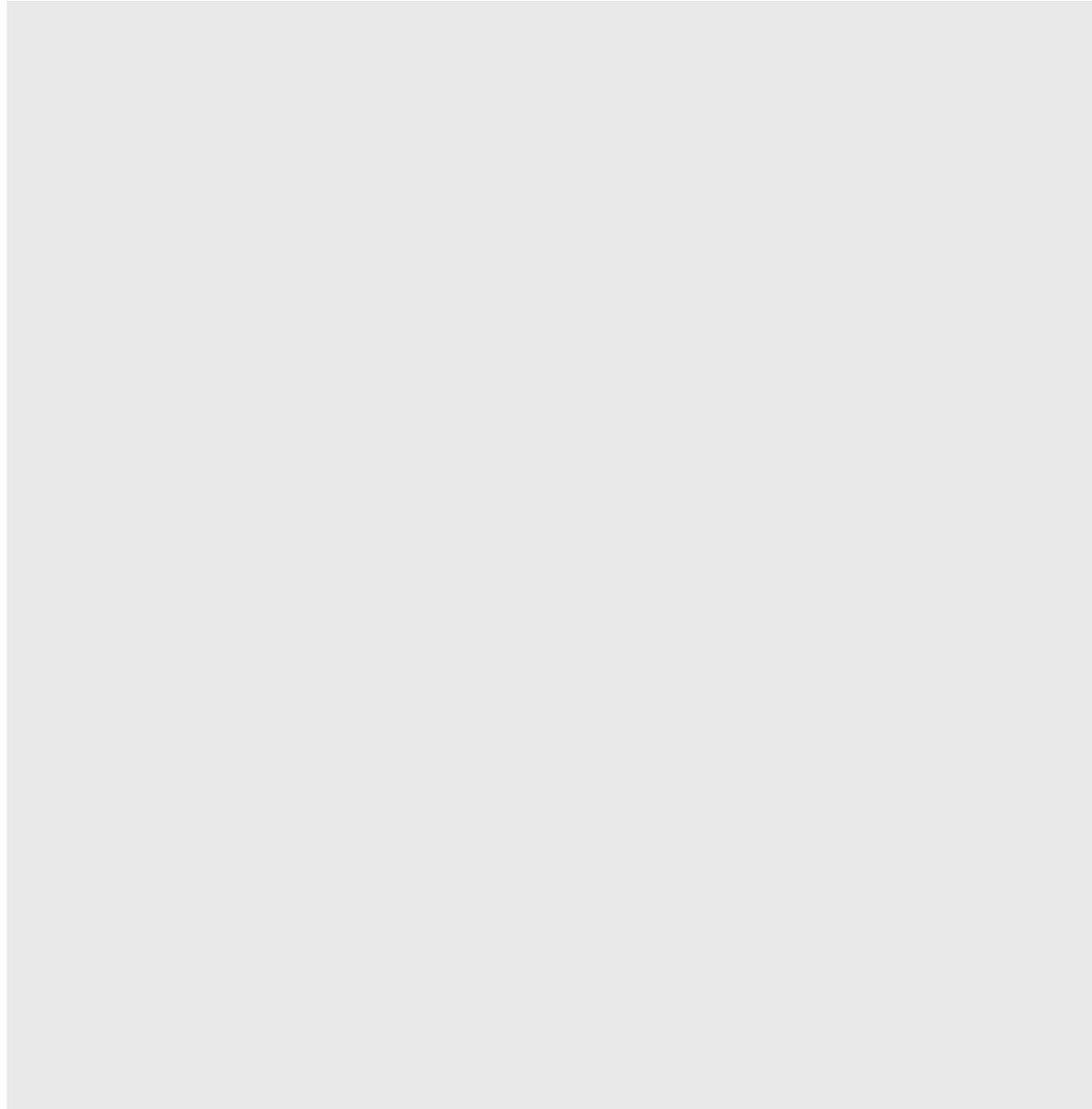
$$\hat{x}_t = \underset{x_t}{\operatorname{argmax}} \{ p(z_t \mid x_t, \hat{m}_{t-1}) \cdot p(x_t \mid u_{t-1}, \hat{x}_{t-1}) \}$$

current measurement

robot motion

map constructed so far

# Mapping using Scan Matching

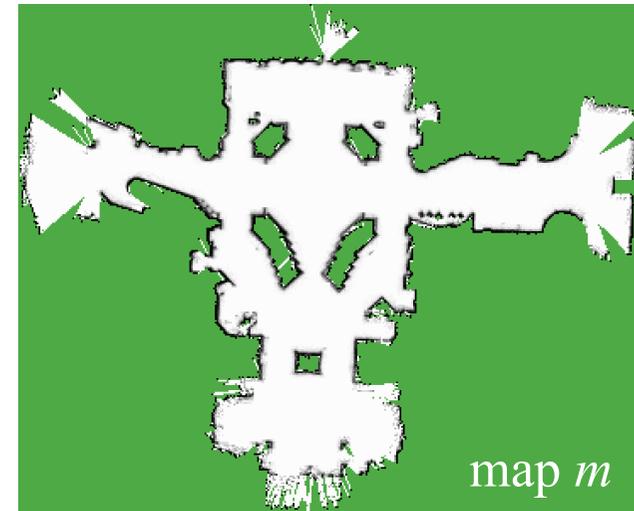


# Probabilistic Formulation of SLAM

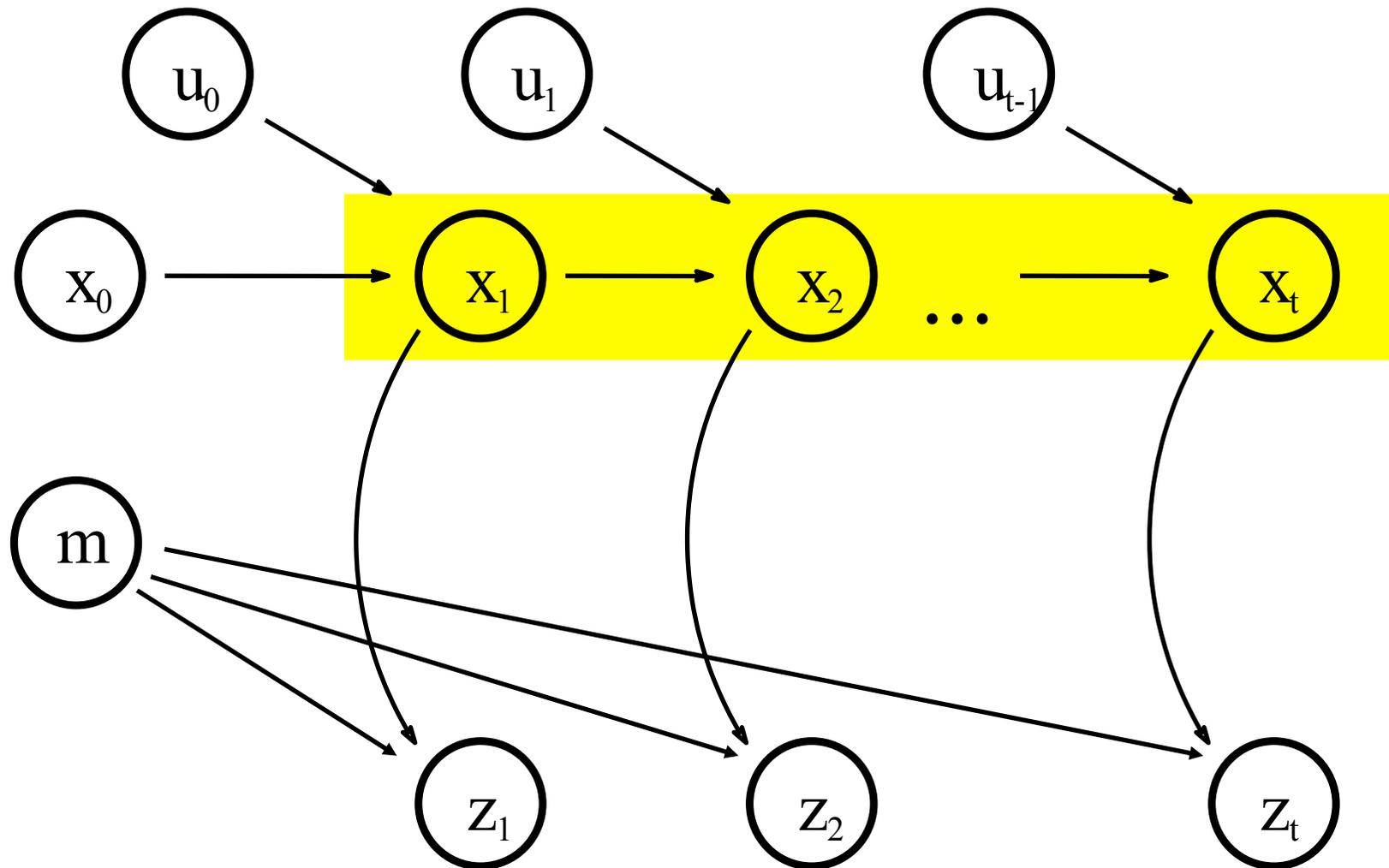
$$Bel(x, m | z, u) = \alpha p(z | x, m) \int_{x'} p(x | u, x') Bel(x', m) dx'$$

$n = a \times b$  dimensions

three dimensions



# A Graphical Model for SLAM



[Murphy et al., 99]

# Rao-Blackwellized Particle Filters for SLAM

## Observation:

Given the true trajectory of the robot, we can efficiently compute the map (mapping with known poses).

## Idea:

- Use a particle filter to represent potential trajectories of the robot.
- Each particle carries its own map.
- Each particle survives with a probability that is proportional to the likelihood of the observation given that particle and its map.

# Factorization Underlying Rao-Blackwellization

$$Bel(x, m | z, u)$$

$$= p(m | x, z, \cancel{u}) p(x | z, u)$$

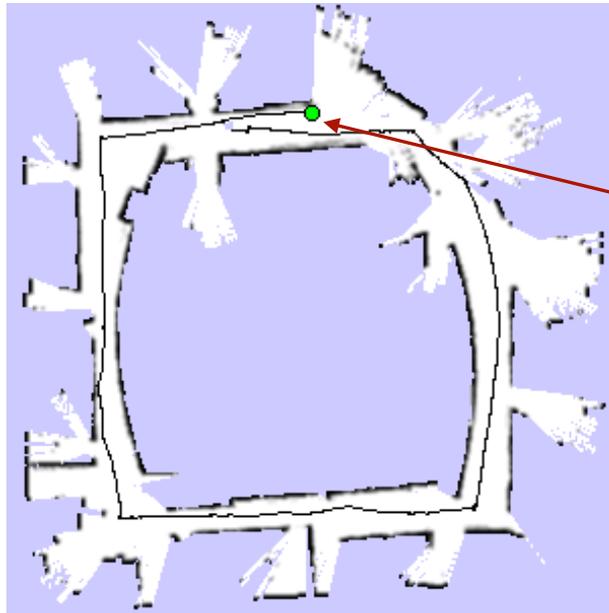
Mapping with known poses



Particle filter representing trajectory hypotheses

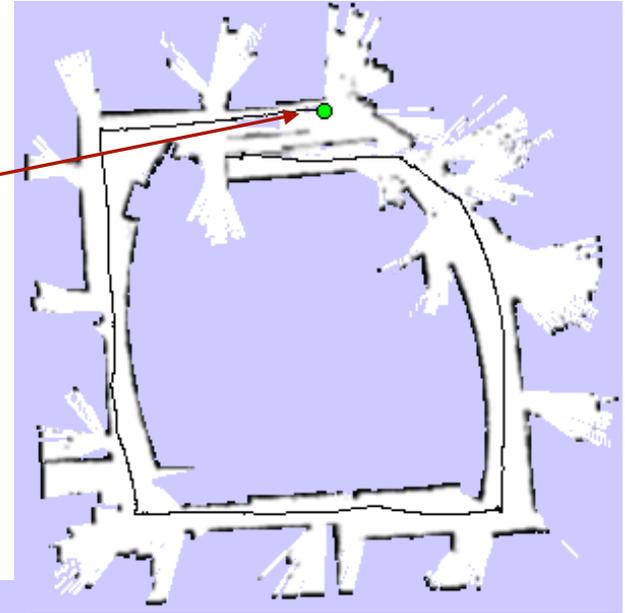
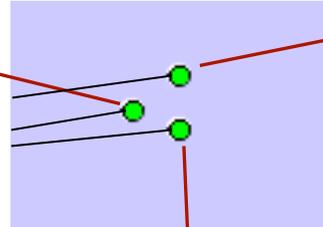


# Particle Filter Realization

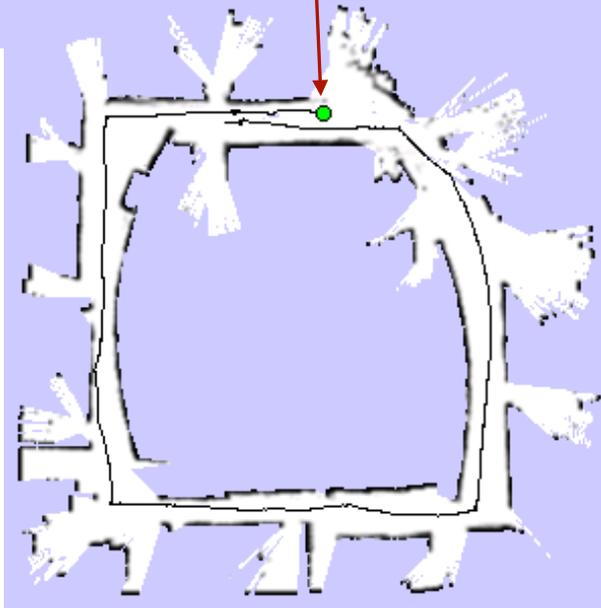


map of particle 1

3 particles



map of particle 3



map of particle 2

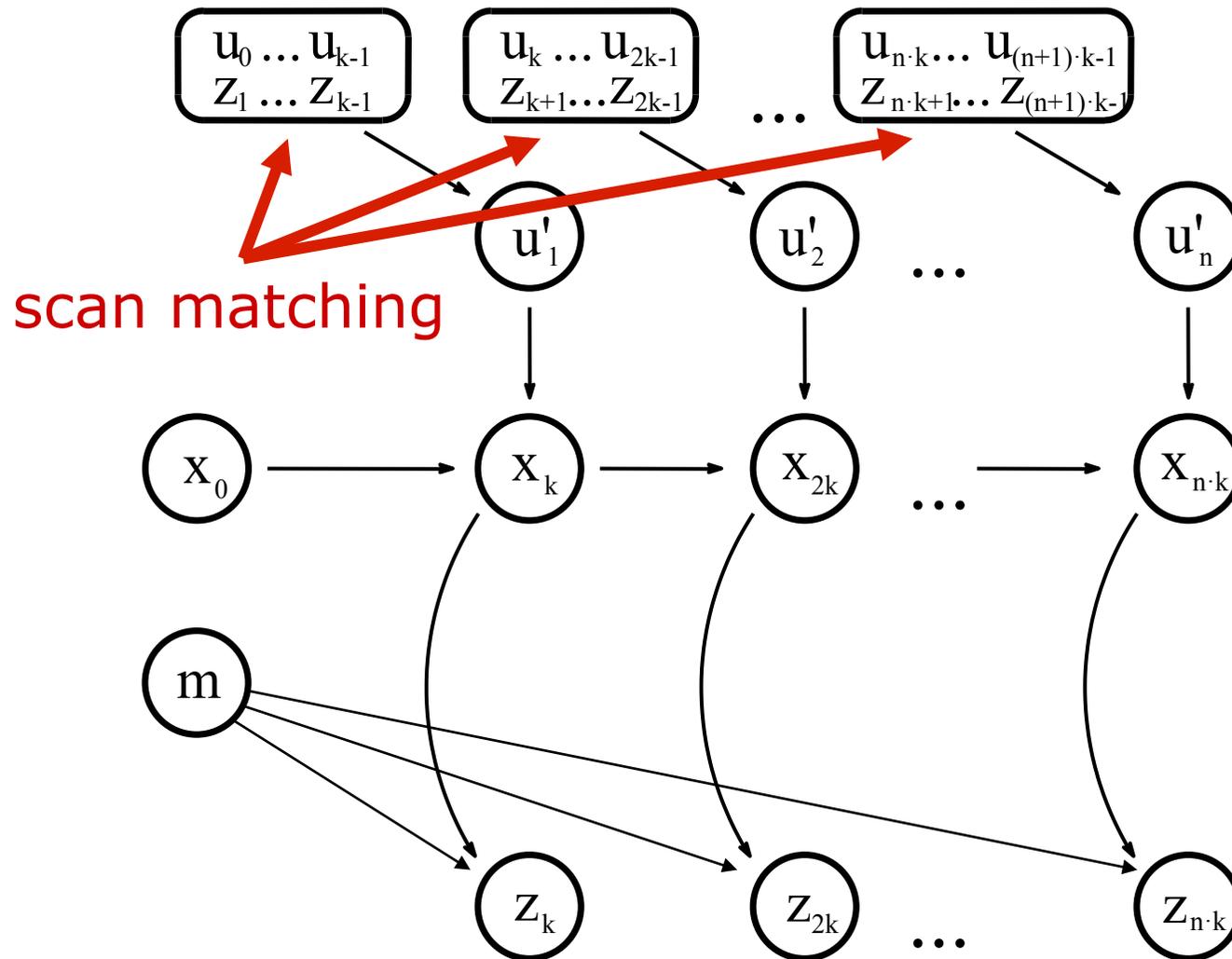
# Challenge

Reduction of the number of particles.

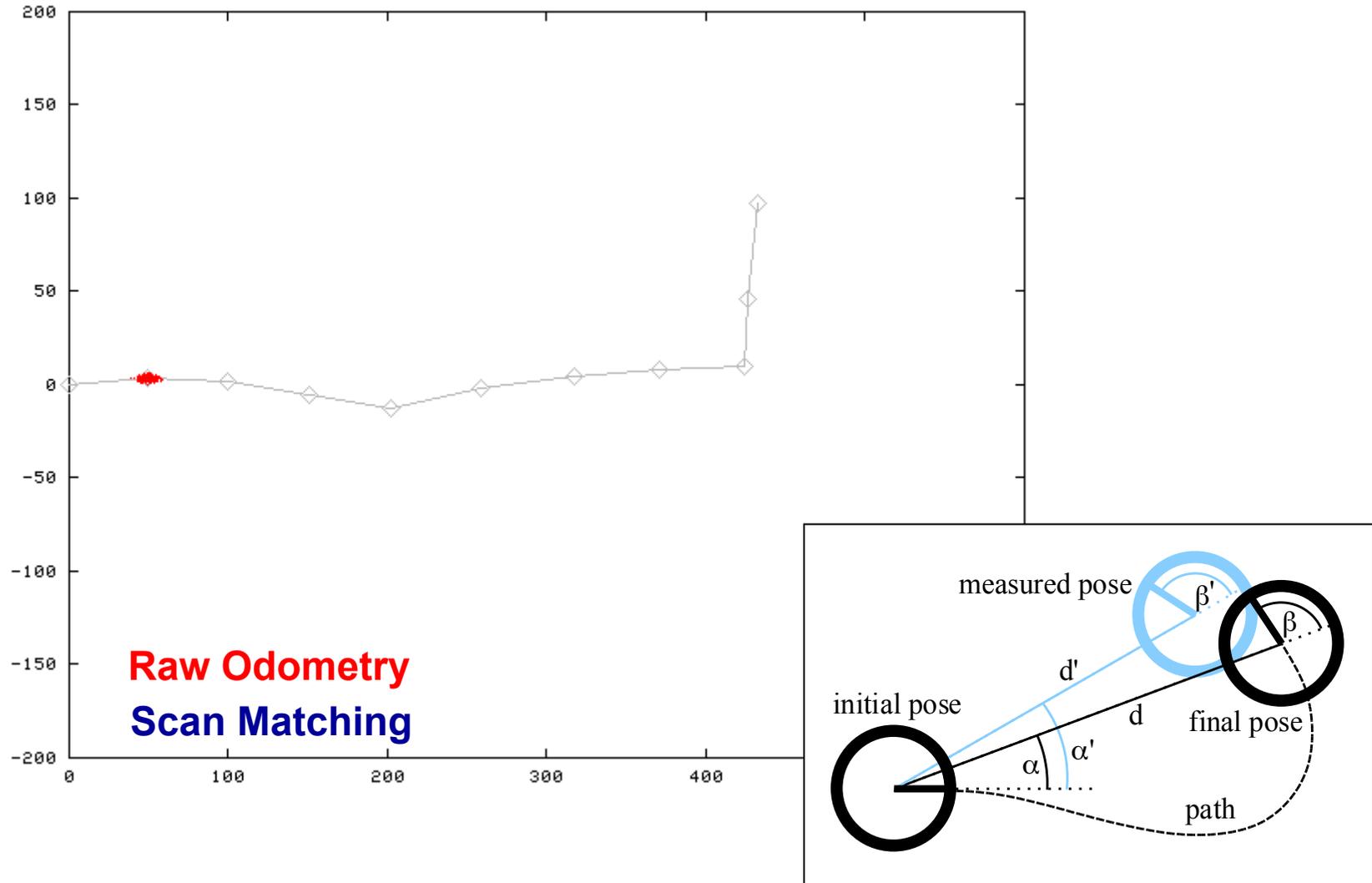
Approaches:

- Focused proposal distributions  
(keep the samples in the right place)
- Adaptive re-sampling  
(avoid depletion of relevant particles)

# Graphical Model for Mapping with Improved Odometry



# Motion Model for Scan Matching

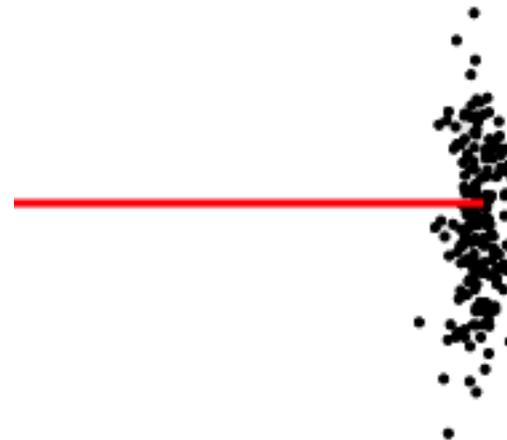


# Incorporating the Current Measurement

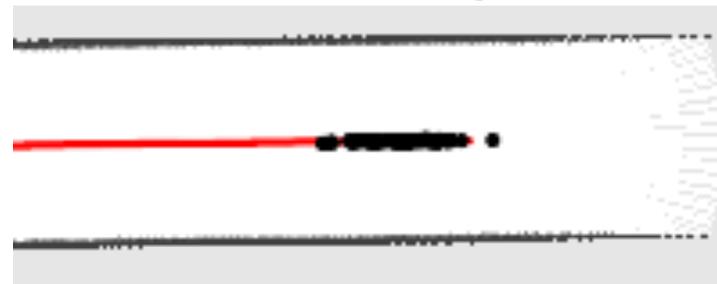
**End of a corridor:**



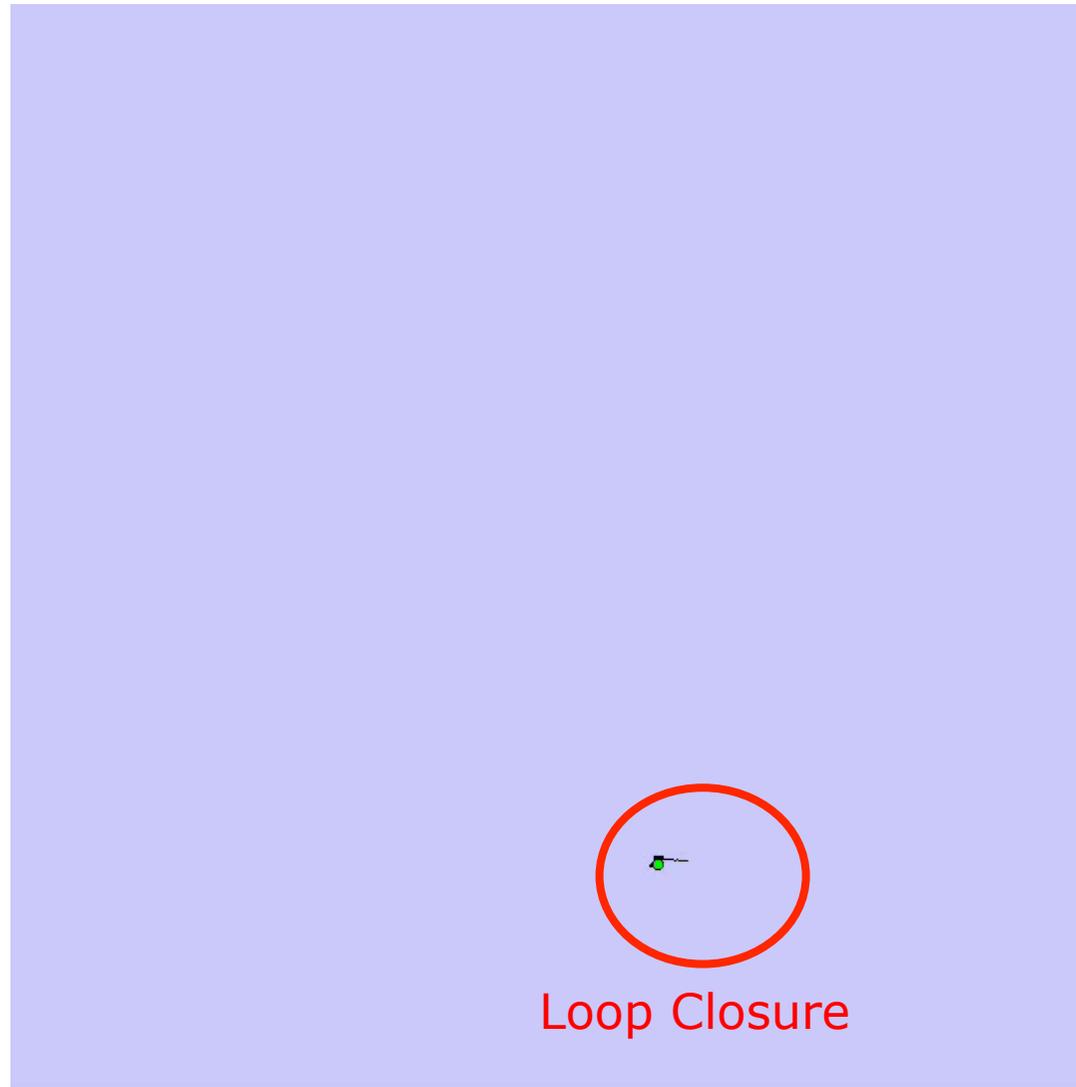
**Free space:**



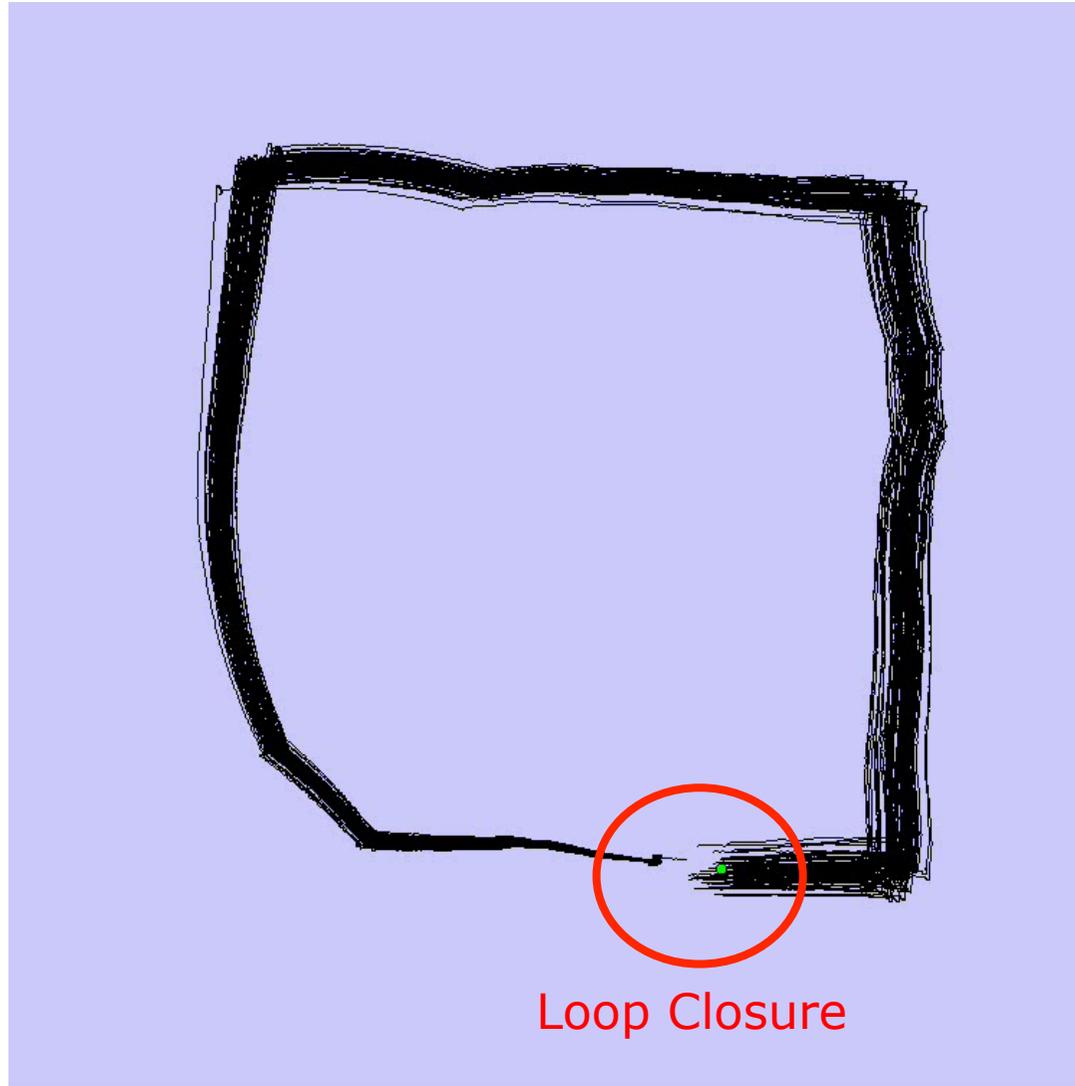
**Corridor:**



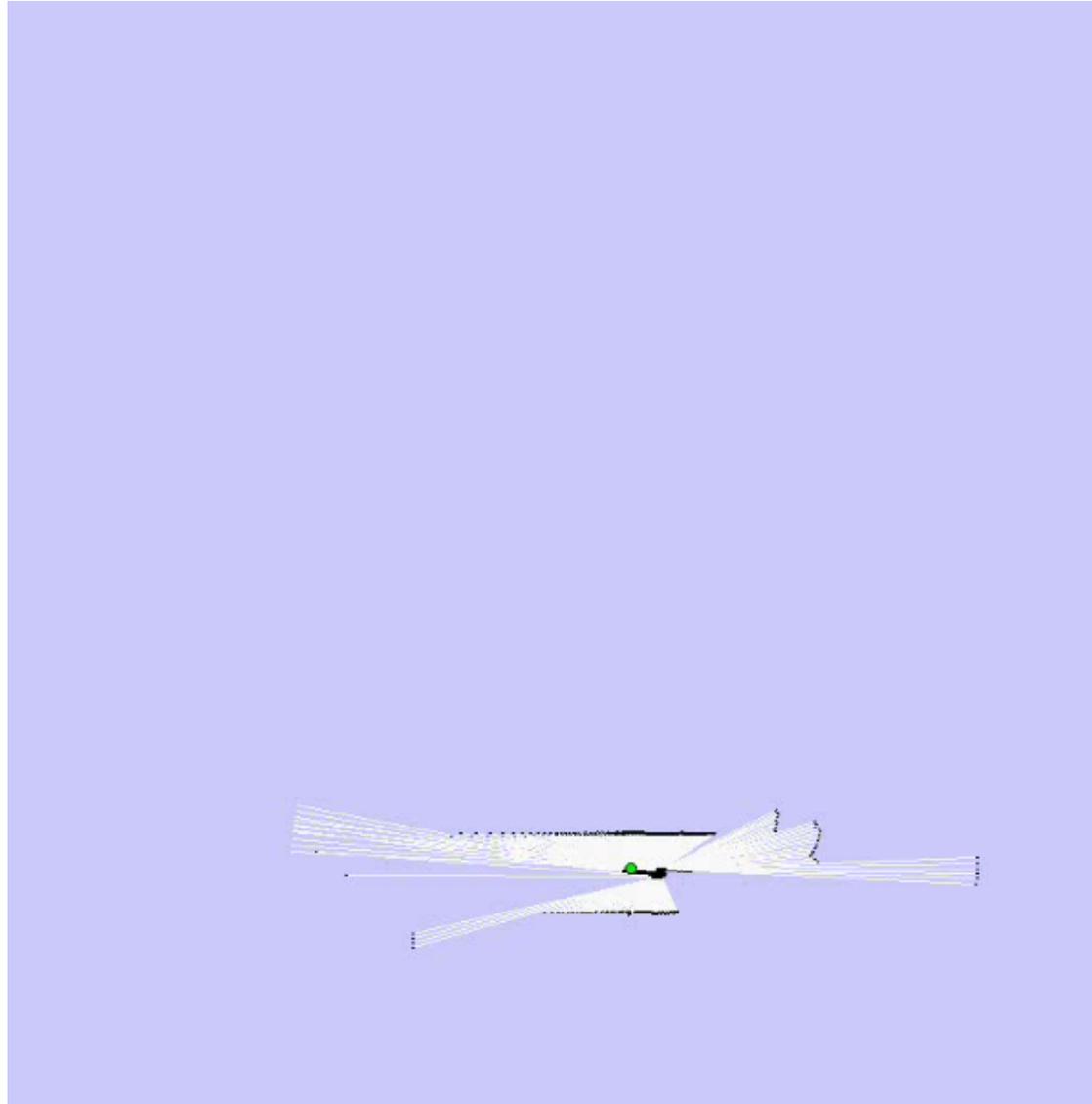
# Application Example



# Application Example



# Application Example

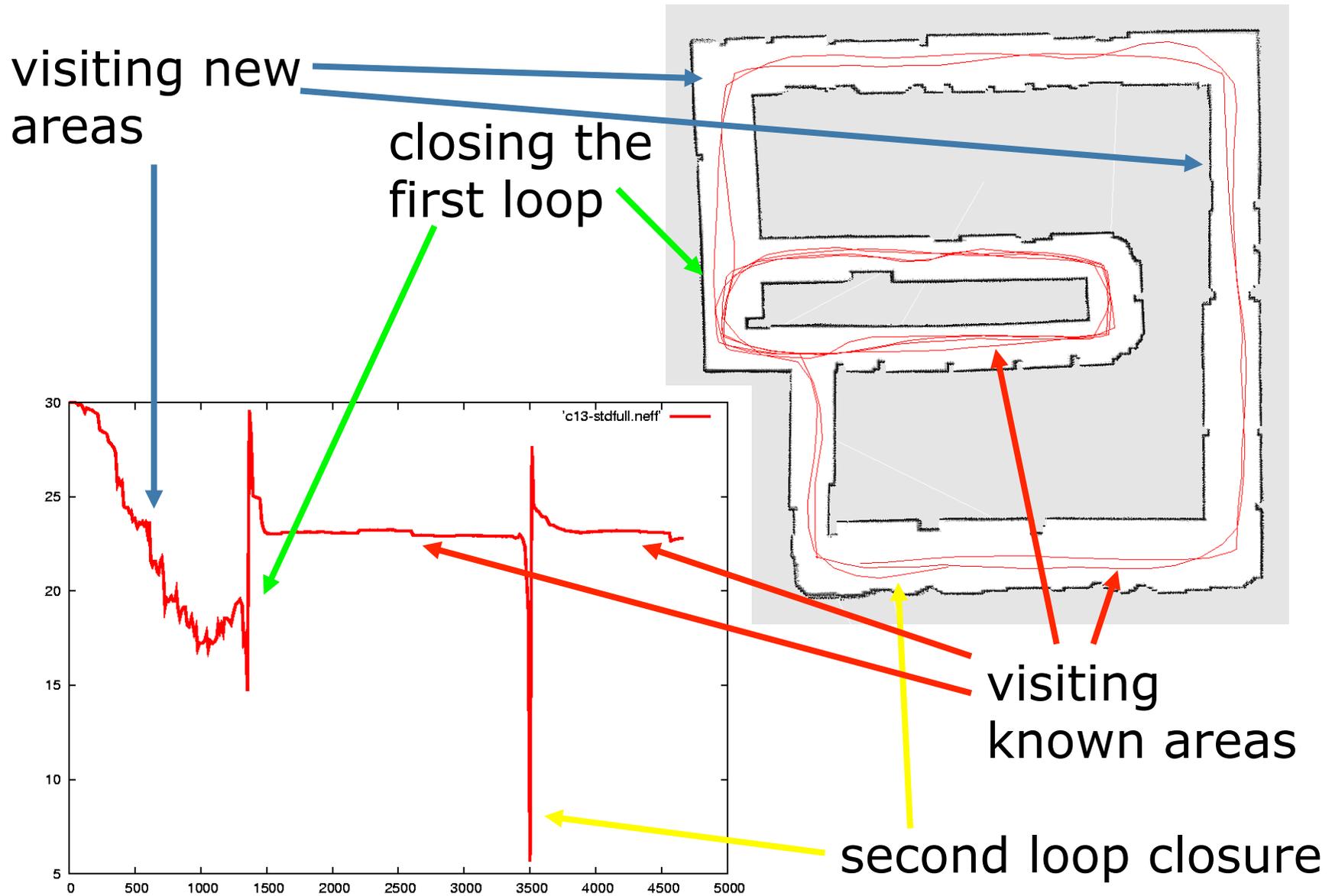


# Number of Effective Particles

$$n_{eff} = \frac{1}{\sum_i \left(w_t^{(i)}\right)^2}$$

- Empirical measure of how well the goal distribution is approximated by samples drawn from the proposal.
- $n_{eff}$  describes “the variance of the particle weights”
- $n_{eff}$  is maximal for equal weights. In this case, the distribution is close to the proposal.
- We only re-sample when  $n_{eff}$  drops below a given threshold ( $n/2$ )

# Typical Evolution of $n_{eff}$

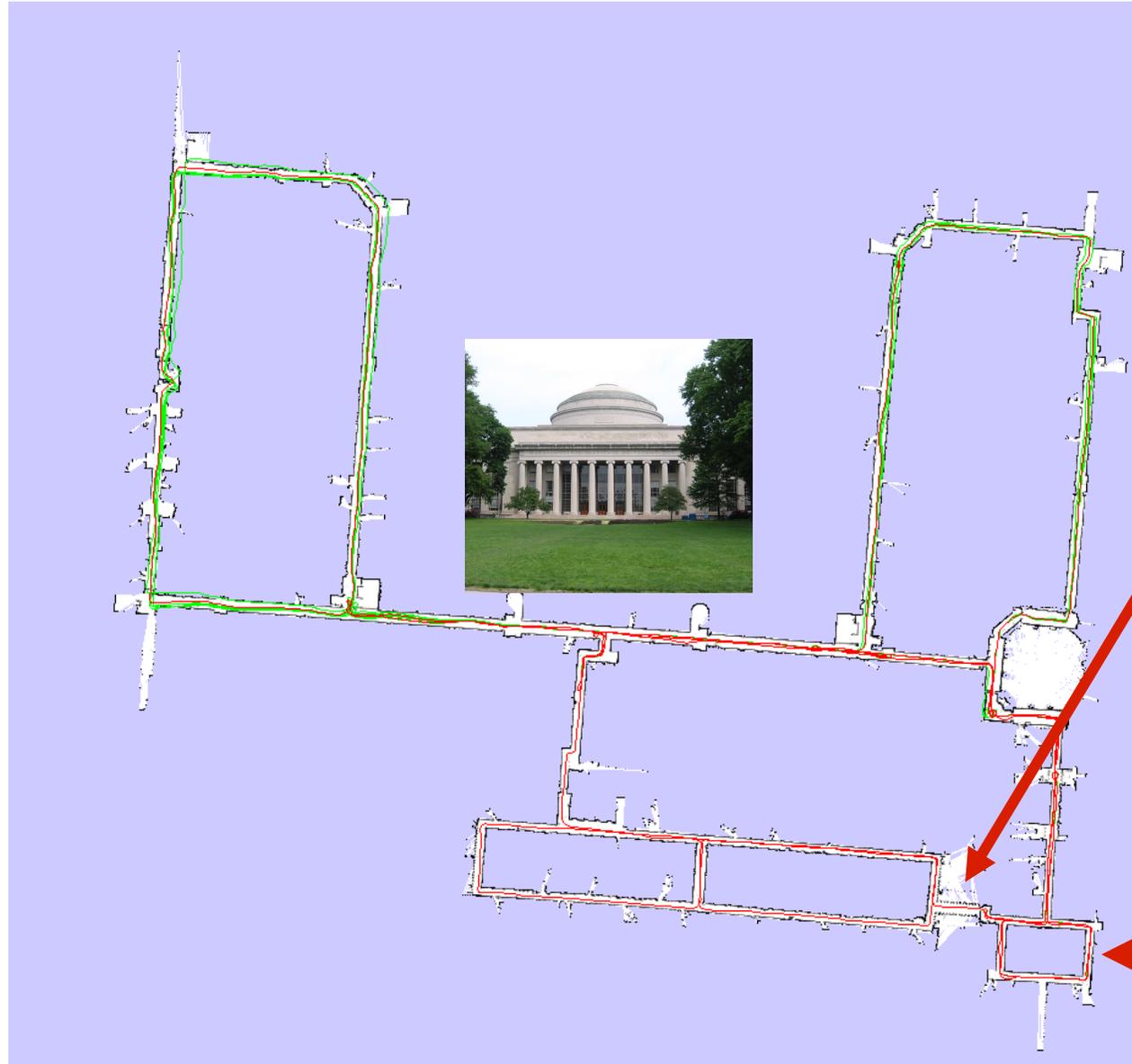


# Map of the Intel Lab



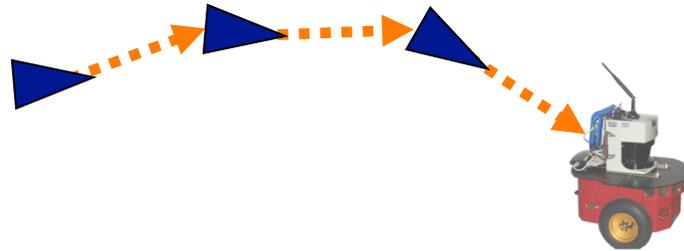
- **15 particles**
- four times faster than real-time P4, 2.8GHz
- 5cm resolution during scan matching
- 1cm resolution in final map

# MIT Killian Court



# Graph-Based SLAM

- Constraints connect the poses of the robot while it is moving
- Constraints are inherently uncertain

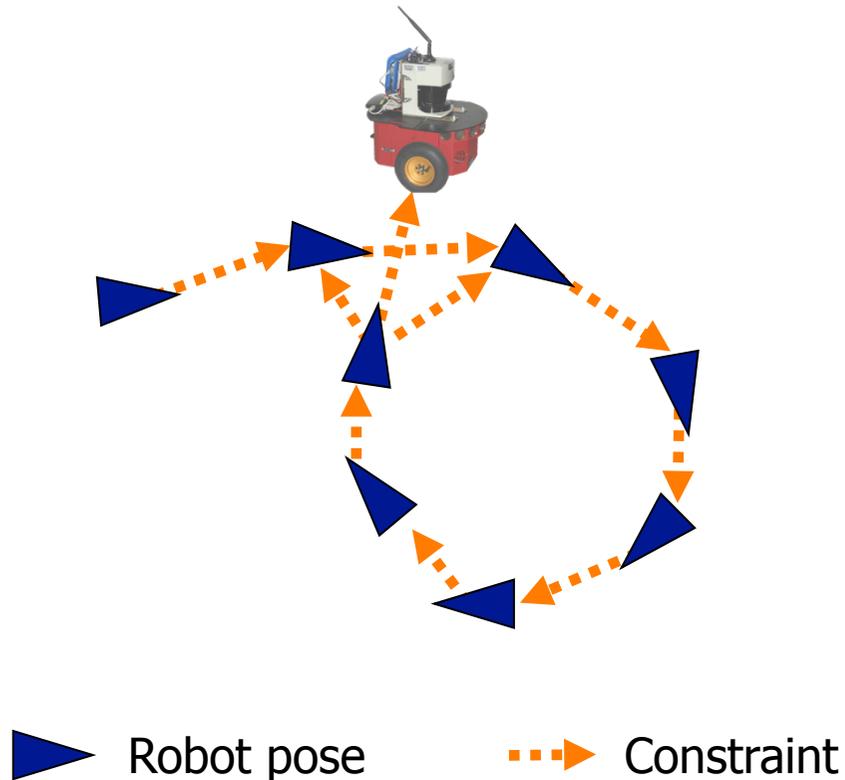


▶ Robot pose

---▶ Constraint

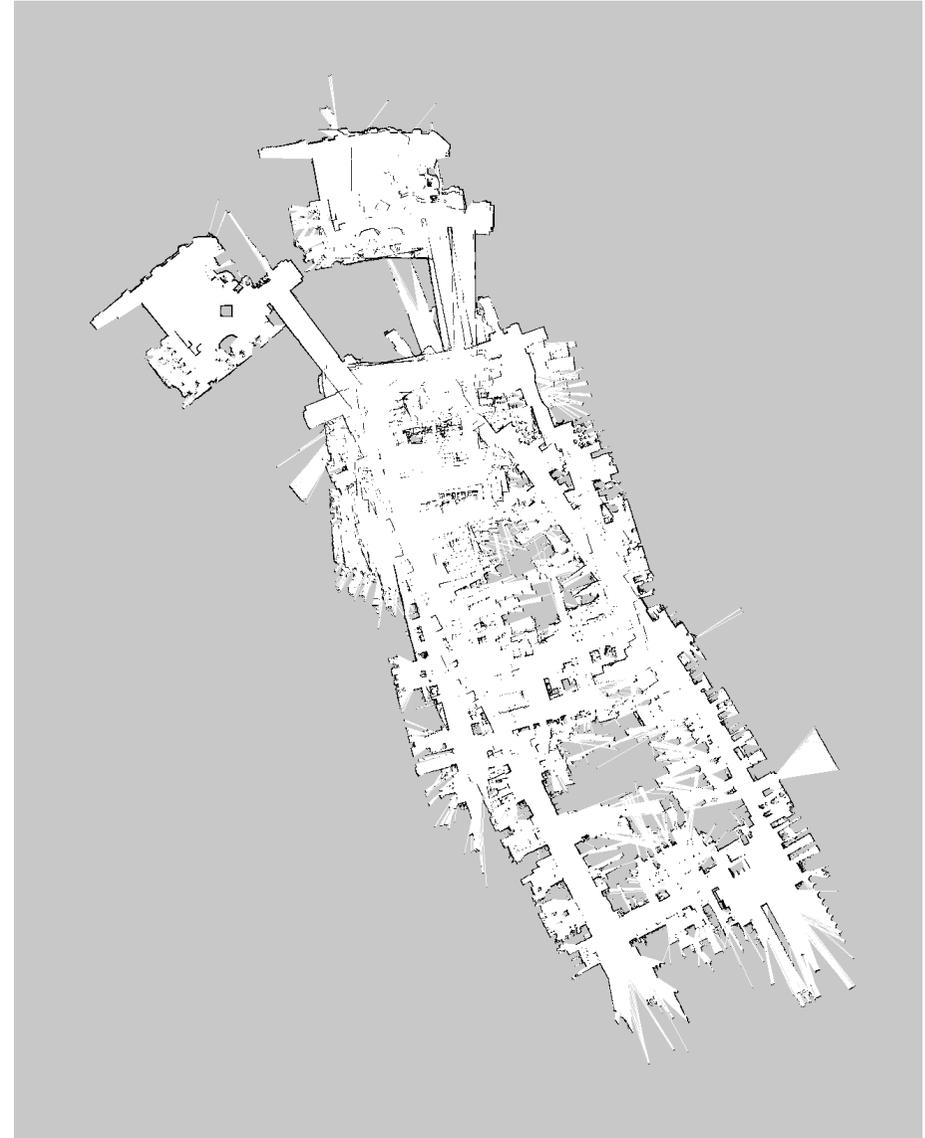
# Graph-Based SLAM

- Observing previously seen areas generates constraints between non-successive poses



# Graph-Based SLAM in a Nutshell

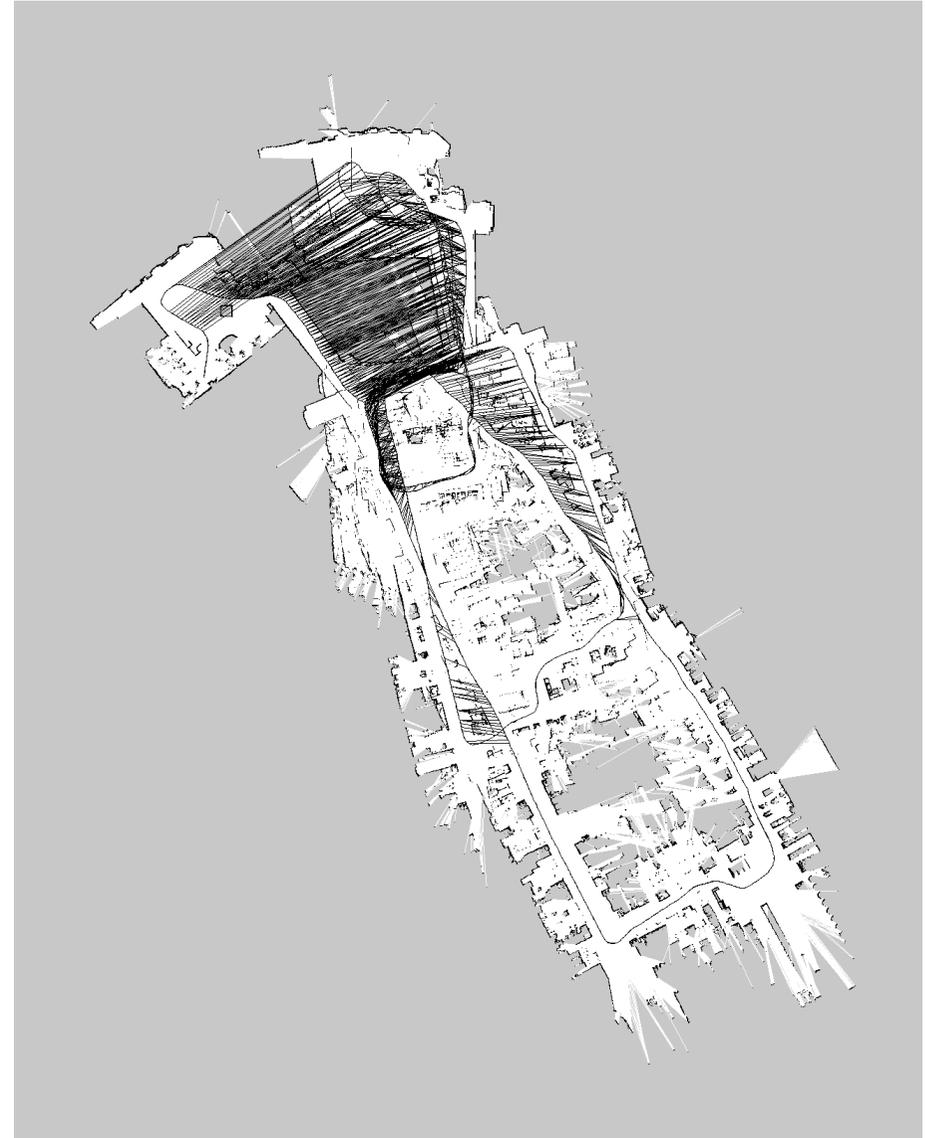
- Every node in the graph corresponds to a robot position and a laser measurement
- An edge between two nodes represents a spatial constraint between the nodes



KUKA Halle 22, courtesy of P. Pfaff

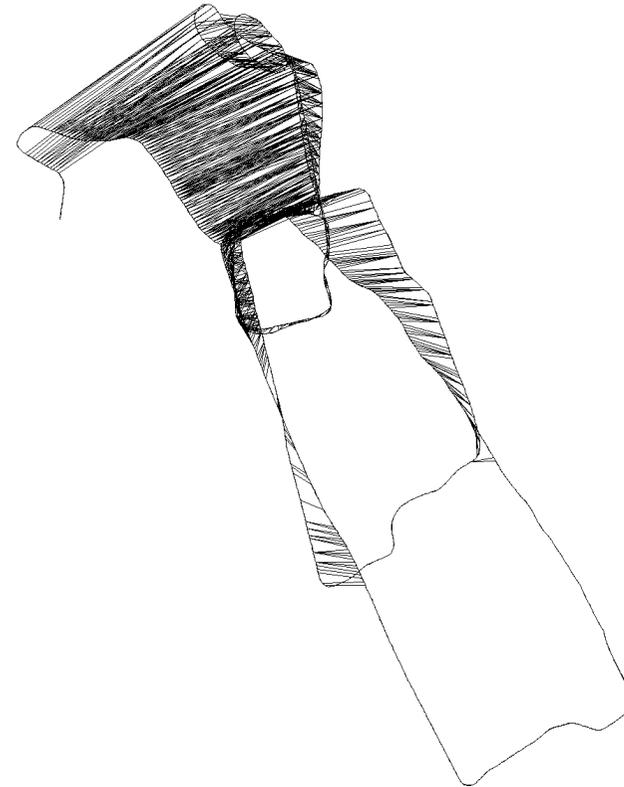
# Graph-Based SLAM in a Nutshell

- Every node in the graph corresponds to a robot position and a laser measurement
- An edge between two nodes represents a spatial constraint between the nodes



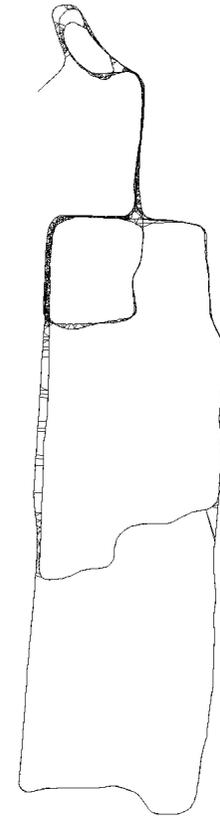
# Graph-Based SLAM in a Nutshell

- Once we have the graph, we determine the most likely map by correcting the nodes



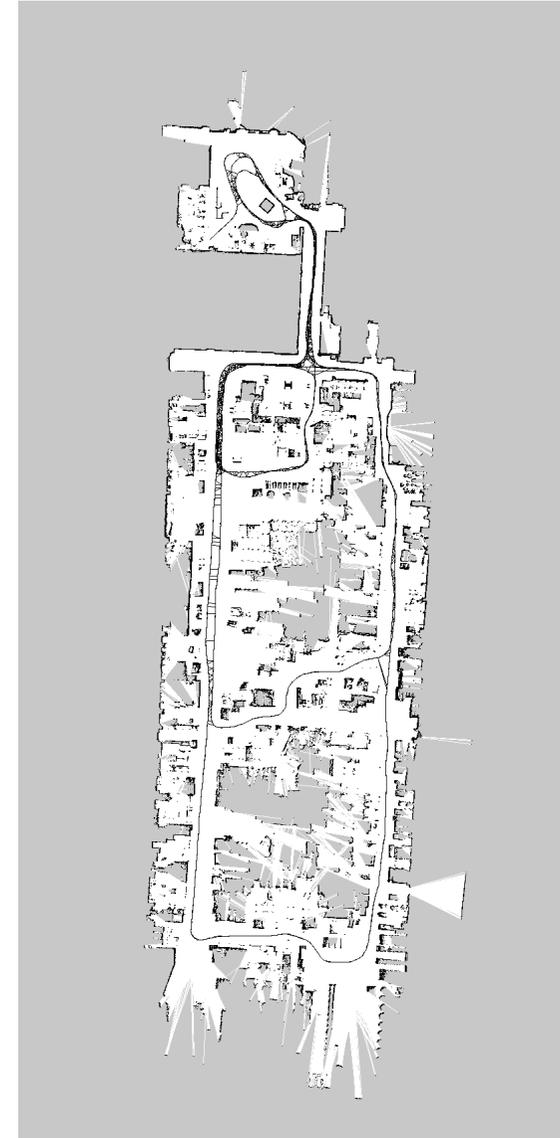
# Graph-Based SLAM in a Nutshell

- Once we have the graph, we determine the most likely map by correcting the nodes  
... like this



# Graph-Based SLAM in a Nutshell

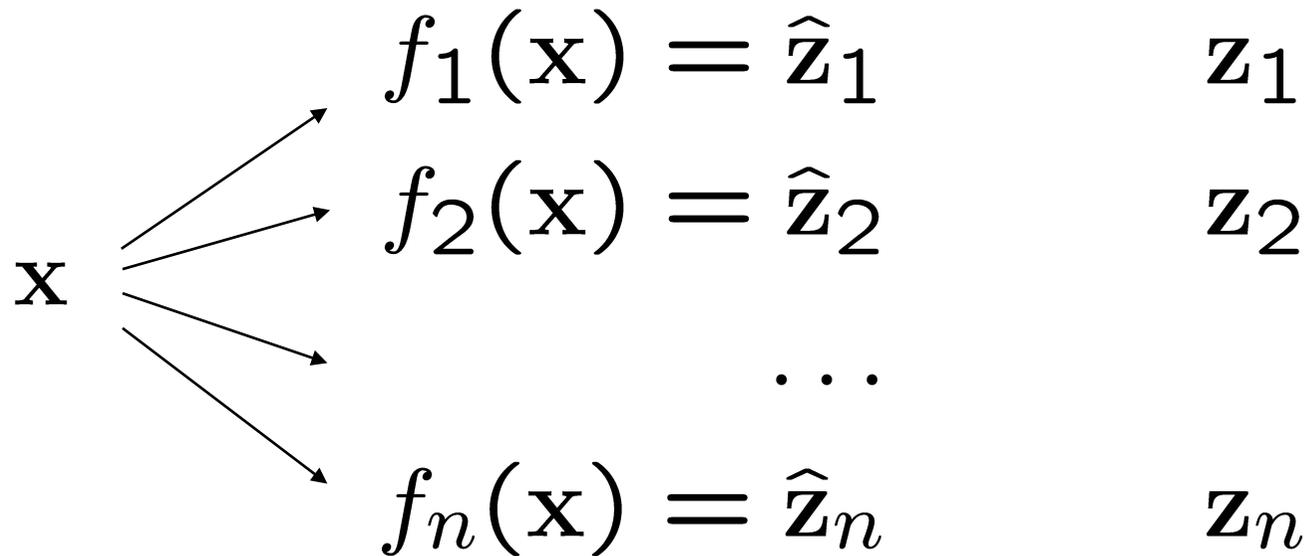
- Once we have the graph, we determine the most likely map by correcting the nodes
  - ... like this
- Then, we can render a map based on the known poses



# Least Squares in General

- Approach for computing a solution for an **overdetermined system**
- “More equations than unknowns”
- Minimizes the **sum of the squared errors** in the equations
- Standard approach to a large set of problems

# Graphical Explanation



state  
(unknown)

predicted  
measurements

real  
measurements

# Error Function

- Error  $\mathbf{e}_i$  is typically the **difference** between the **predicted and actual** measurement

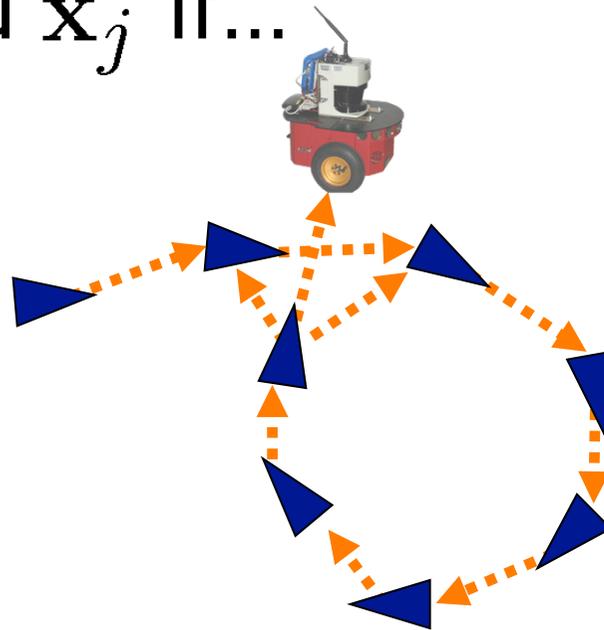
$$\mathbf{e}_i(\mathbf{x}) = \mathbf{z}_i - f_i(\mathbf{x})$$

- We assume that the error has **zero mean** and is **normally distributed**
- Gaussian error with information matrix  $\mathbf{\Omega}_i$
- The squared error of a measurement depends only on the state and is a scalar

$$e_i(\mathbf{x}) = \mathbf{e}_i(\mathbf{x})^T \mathbf{\Omega}_i \mathbf{e}_i(\mathbf{x})$$

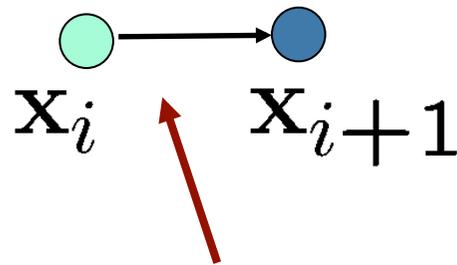
# The Graph

- It consists of  $n$  nodes  $\mathbf{x} = \mathbf{x}_{1:n}$
- Each  $\mathbf{x}_i$  is a 2D or 3D transformation (the pose of the robot at time  $t_i$ )
- A constraint/edge exists between the nodes  $\mathbf{x}_i$  and  $\mathbf{x}_j$  if...



# Create an Edge If... (1)

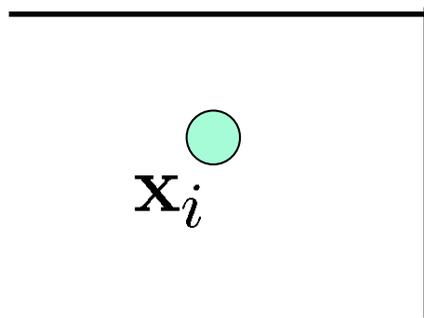
- ...the robot moves from  $x_i$  to  $x_{i+1}$
- Edge corresponds to odometry



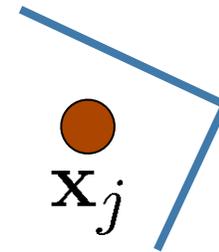
The edge represents the **odometry** measurement

## Create an Edge If... (2)

- ...the robot observes the same part of the environment from  $x_i$  and from  $x_j$
- Construct a **virtual measurement** about the position of  $x_j$  seen from  $x_i$



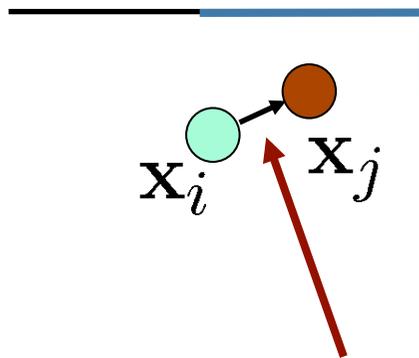
Measurement from  $x_i$



Measurement from  $x_j$

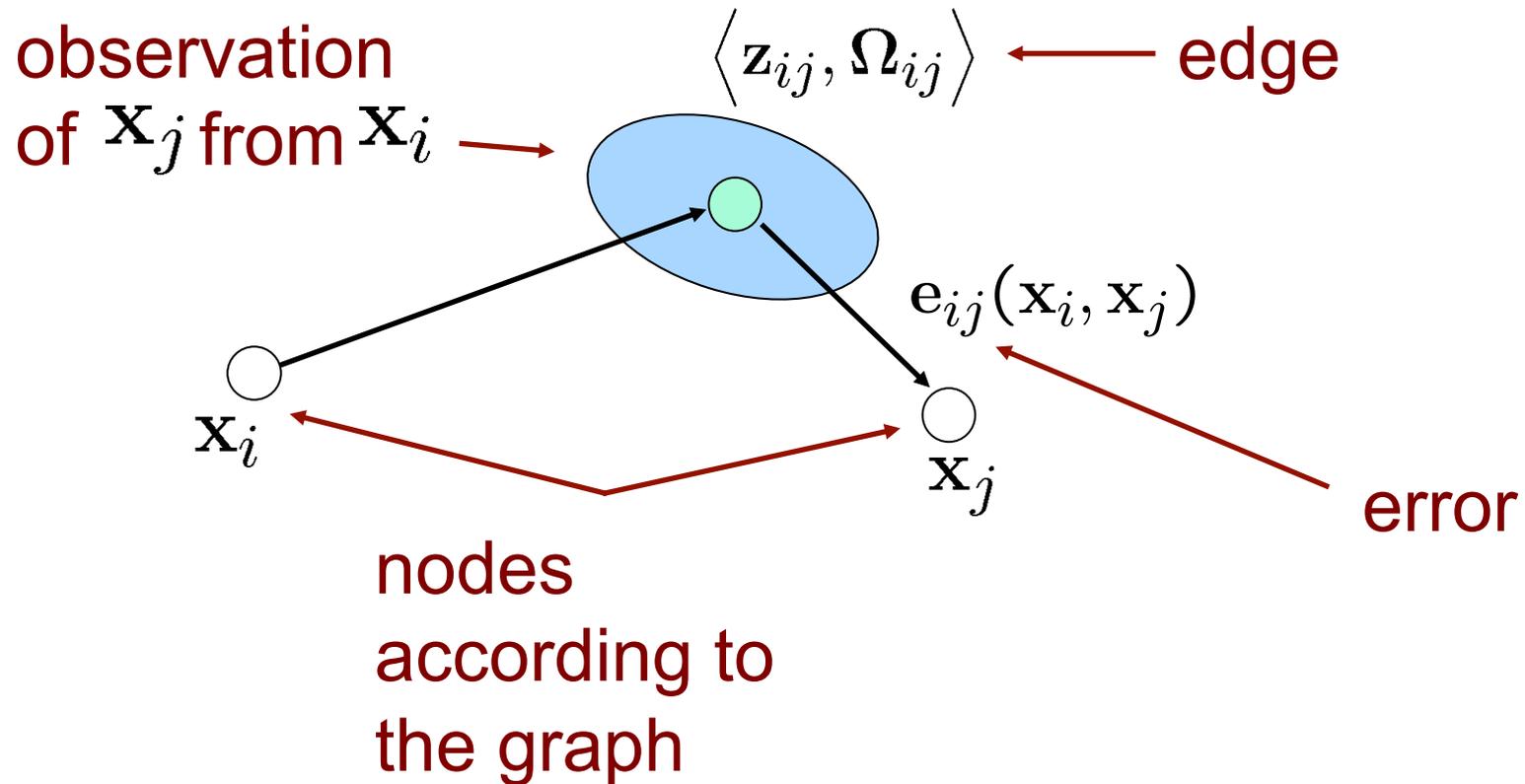
## Create an Edge If... (2)

- ...the robot observes the same part of the environment from  $x_i$  and from  $x_j$
- Construct a **virtual measurement** about the position of  $x_j$  seen from  $x_i$



Edge represents the position of  $x_j$  seen from  $x_i$  based on the **observation**

# Pose Graph

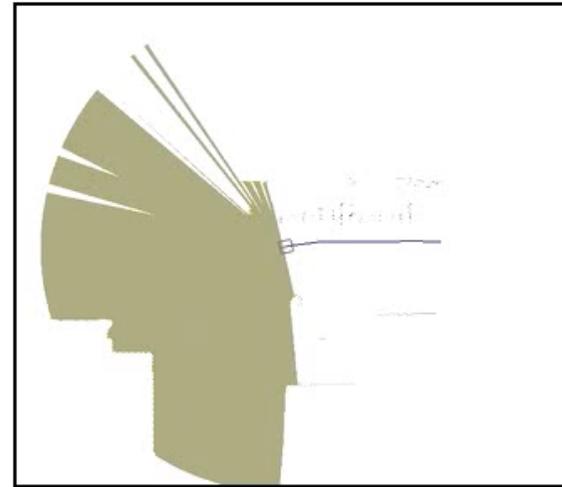


- **Goal:**  $\mathbf{x}^* = \operatorname{argmin}_{\mathbf{x}} \sum_{ij} \mathbf{e}_{ij}^T \Omega_{ij} \mathbf{e}_{ij}$

# Gauss-Newton: The Overall Error Minimization Procedure

- Define the error function
- Linearize the error function
- Compute its derivative
- Set the derivative to zero
- Solve the linear system
- Iterate this procedure until convergence

# Example: CS Campus Freiburg

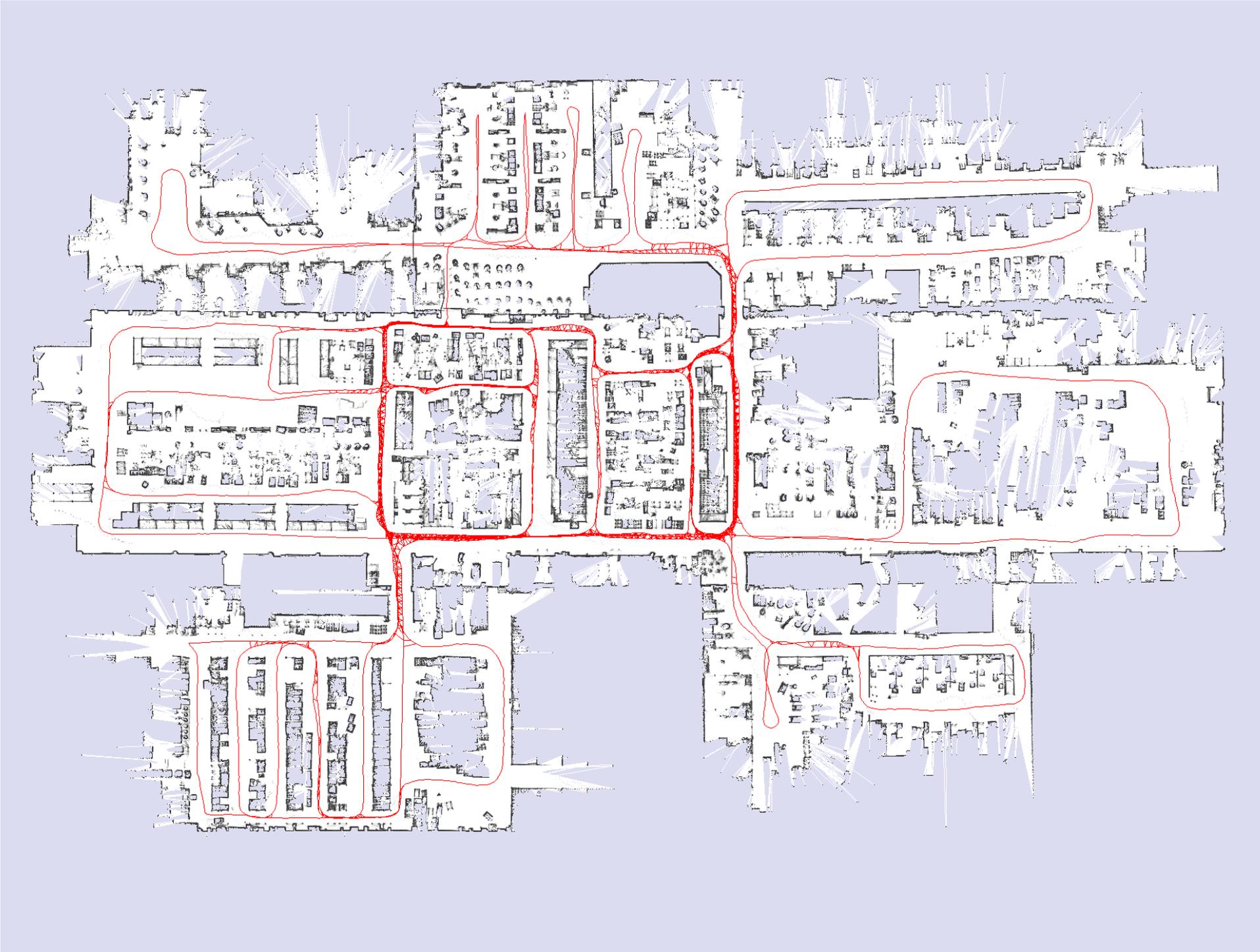


# Application: Sparse Pose Adjustment

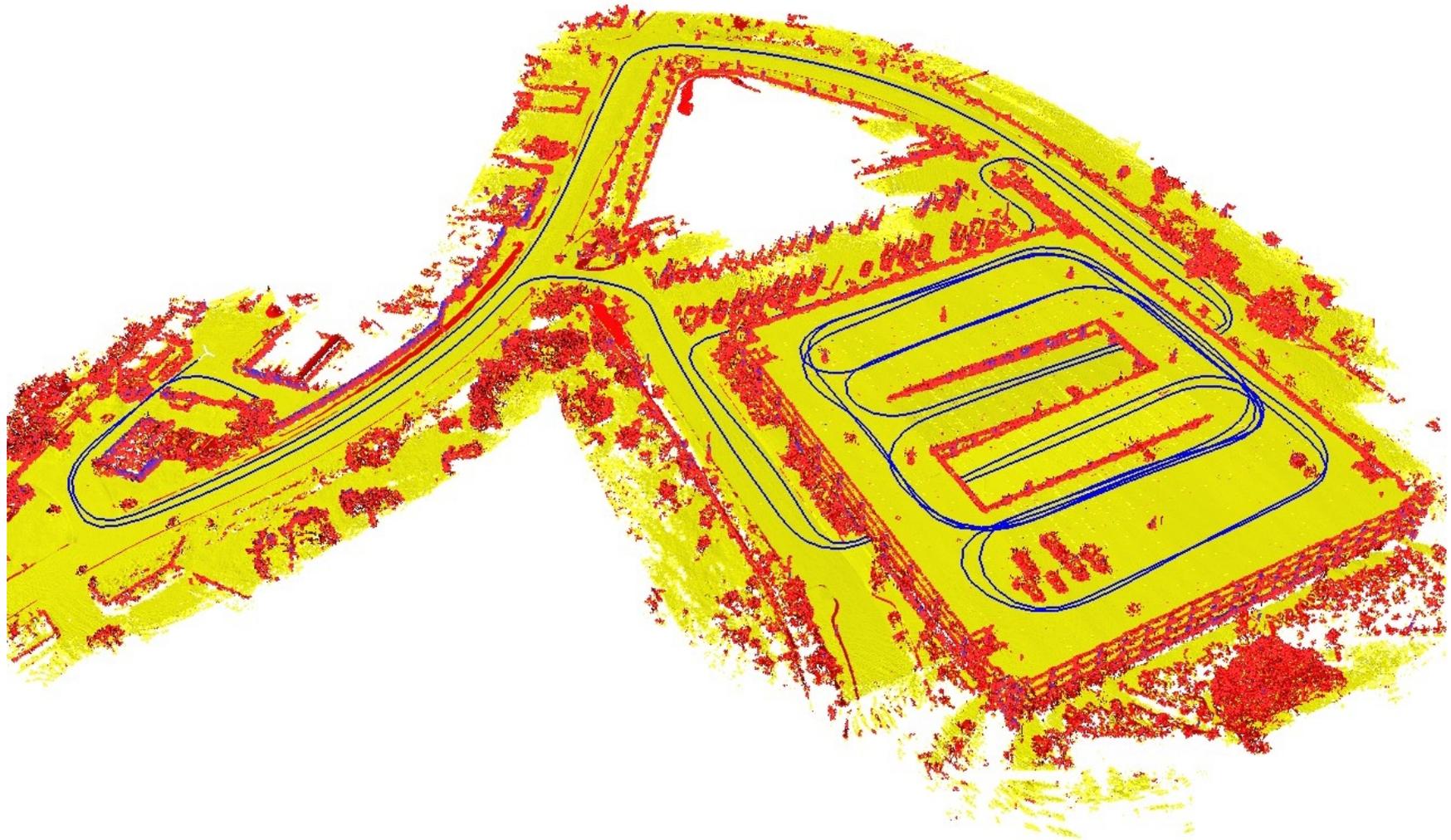


# The KUKA Production Site





# 3D Map of the Stanford Parking Garage



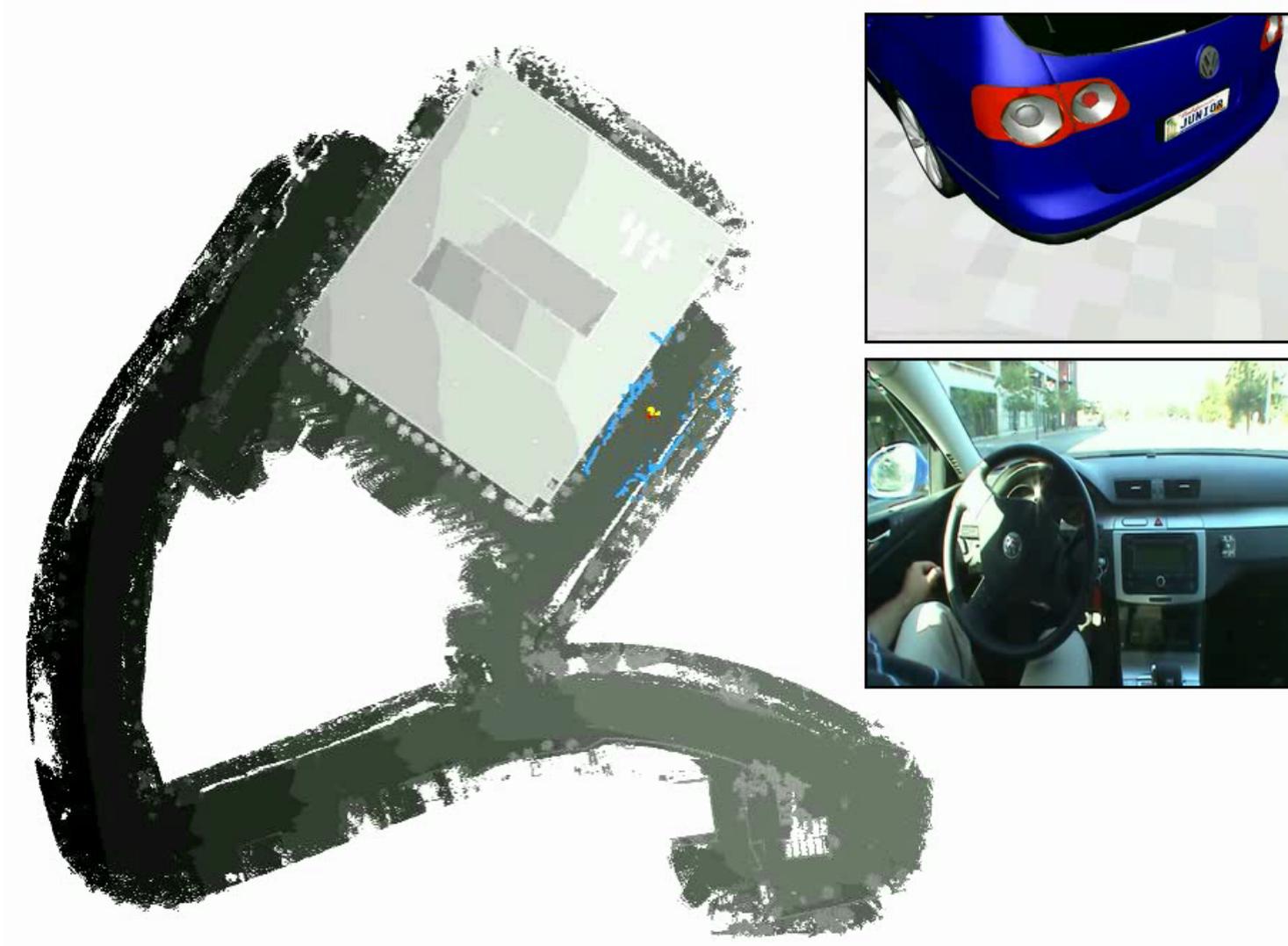
approx. 260MB

# Application: Navigation with the Autonomous Car Junior

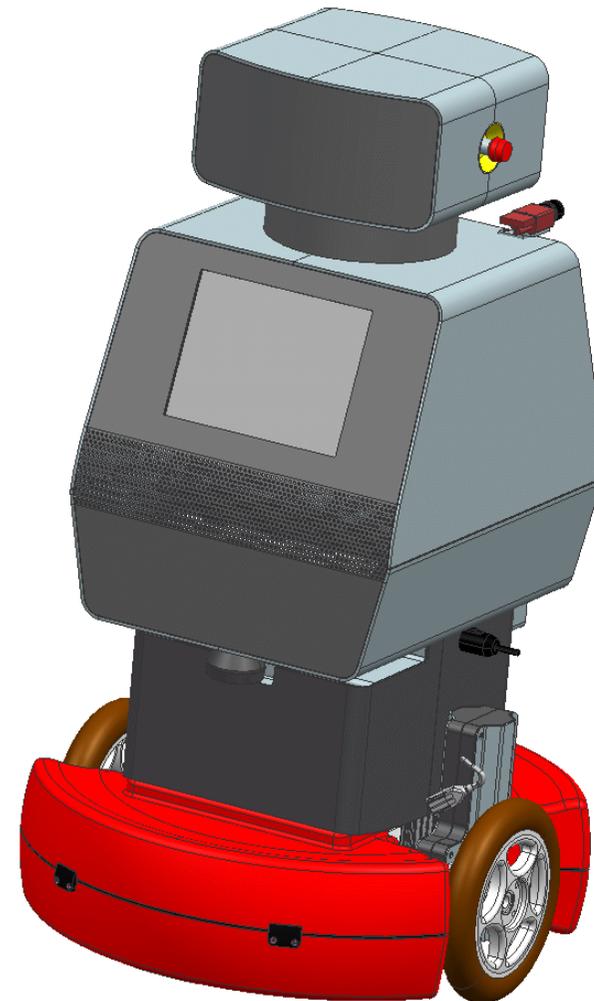
- Task: reach a parking spot on the upper level of the garage.



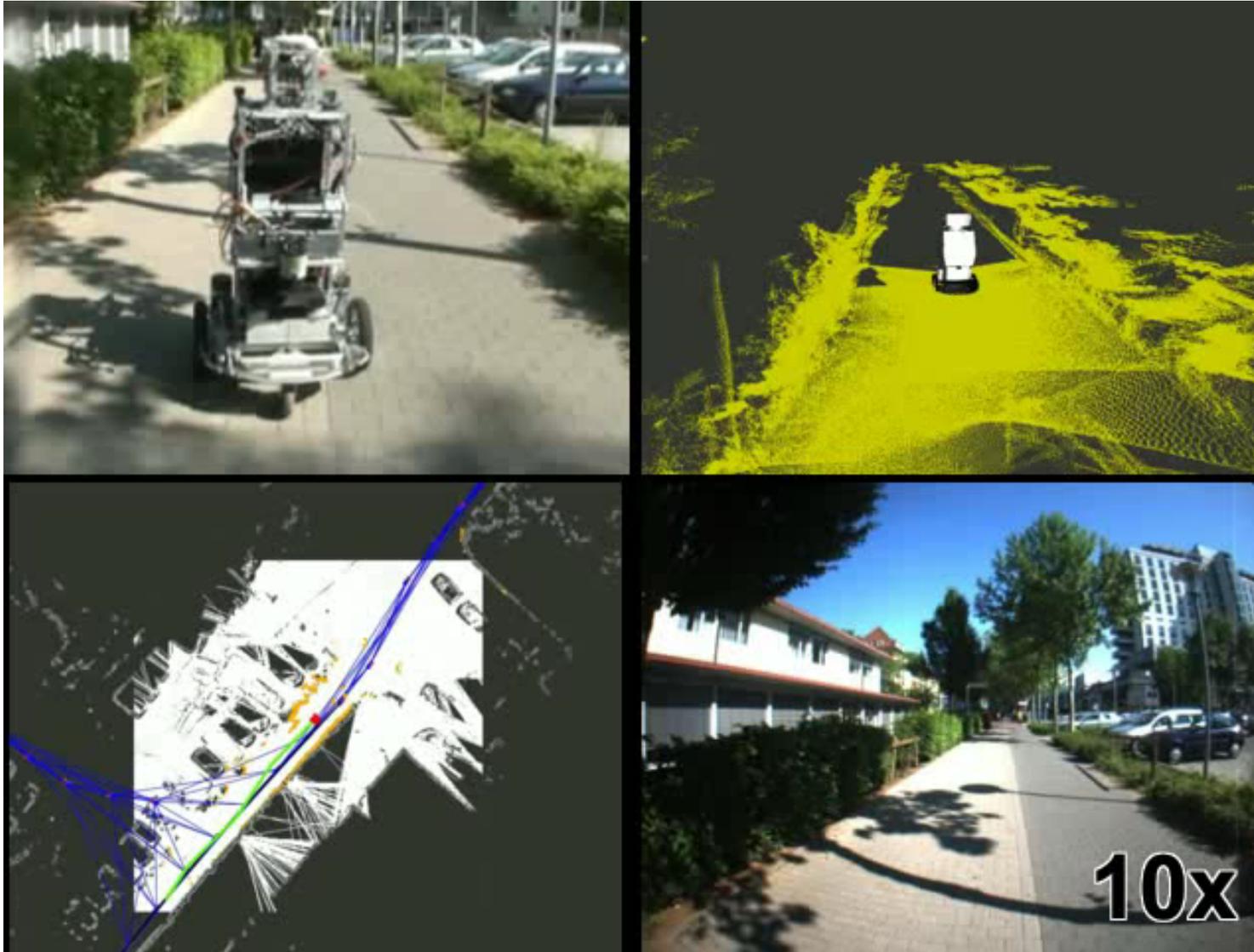
# Autonomous Parking



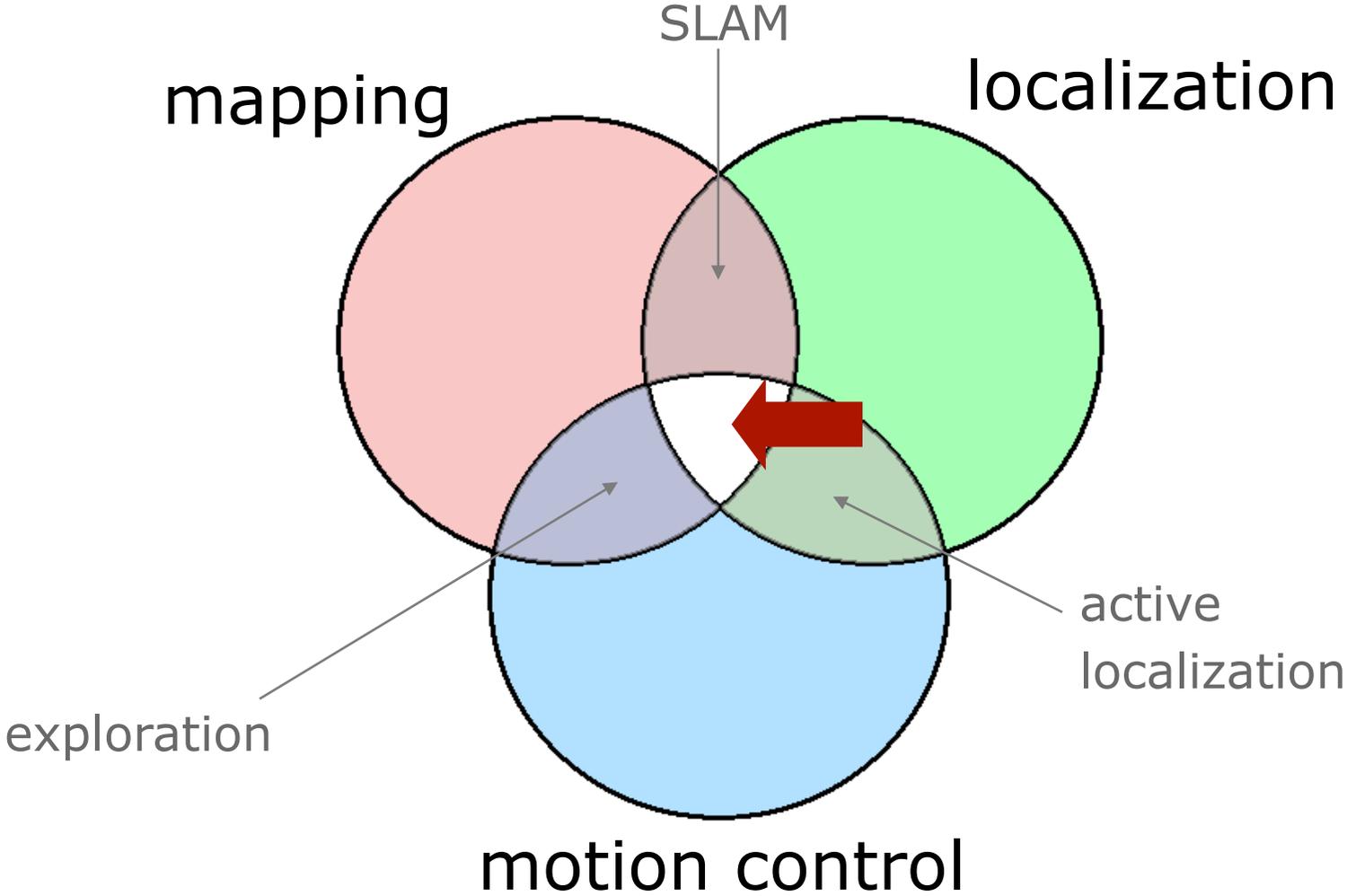
# The Robot



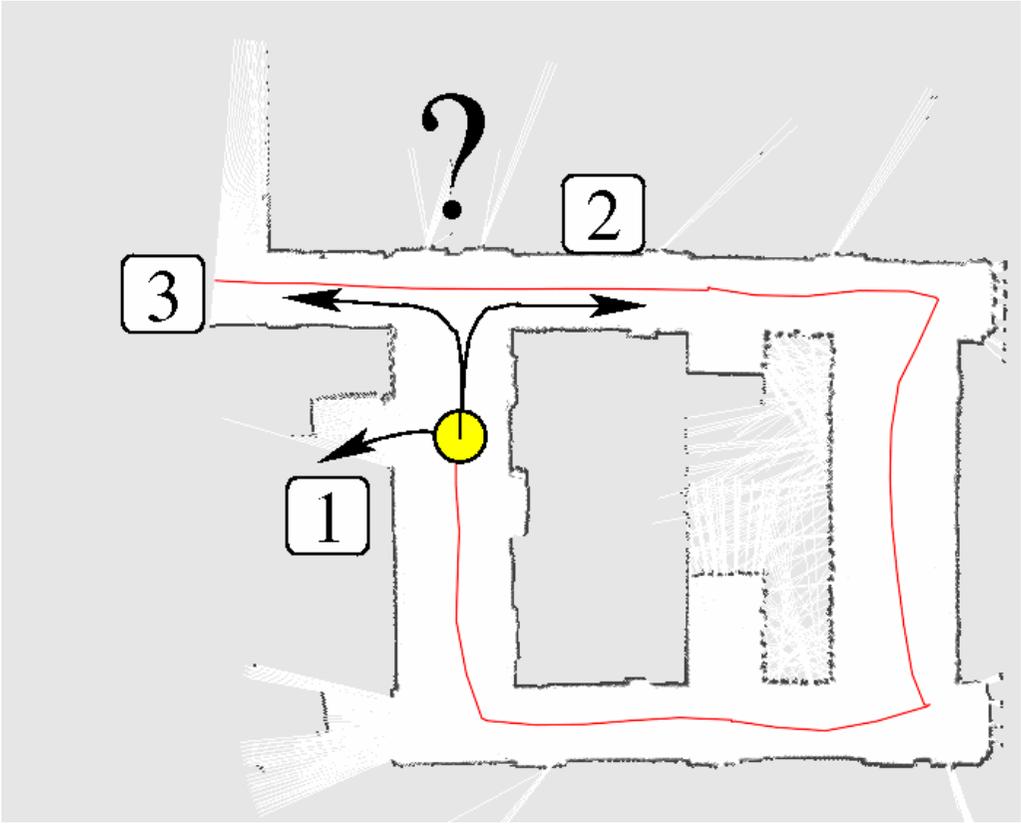
# Navigation in Urban Scenes



# Dimensions of Mobile Robot Navigation



# Which Action to Take?



# Naïve Approach to Combine Exploration and Mapping

Learn the map using a Rao-Blackwellized particle filter.

Apply an exploration approach that minimizes the map uncertainty.

# Disadvantage of the Naïve Approach

Exploration techniques only consider the map uncertainty for generating controls.



They avoid re-visiting known areas.



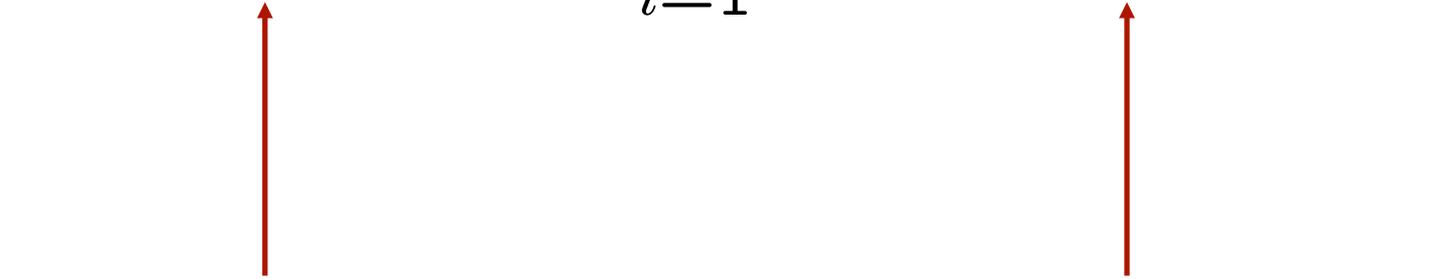
Data association becomes harder.



More particles are needed to learn a correct map.

# Effectively Calculating the Map and Pose Uncertainty

$$\begin{aligned} & H(p(x, m \mid d)) \\ &= H(p(x \mid d)) + \int_x p(x \mid d) H(p(m \mid x, d)) dx \\ &\approx H(p(x \mid d)) + \overset{\#particles}{\sum_{i=1}} \omega^{[i]} H(p(m^{[i]} \mid x^{[i]}, d)) \end{aligned}$$



pose uncertainty                      map uncertainty

# Goal

Integrated approach that considers

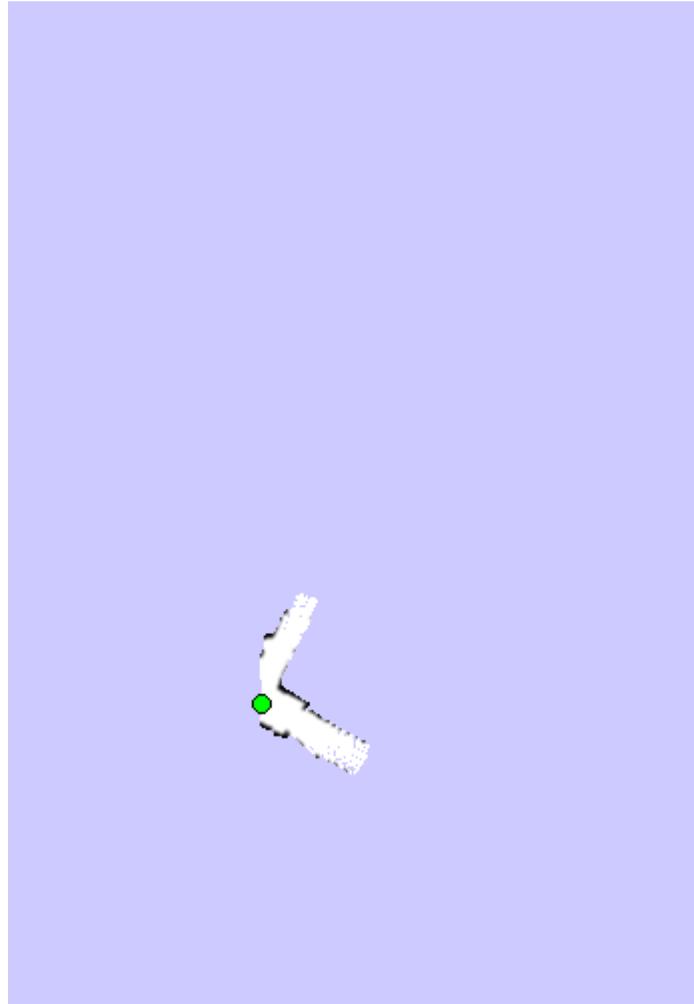
- **exploratory actions,**
- **place revisiting actions,** and
- **loop closing actions**

to control the robot.

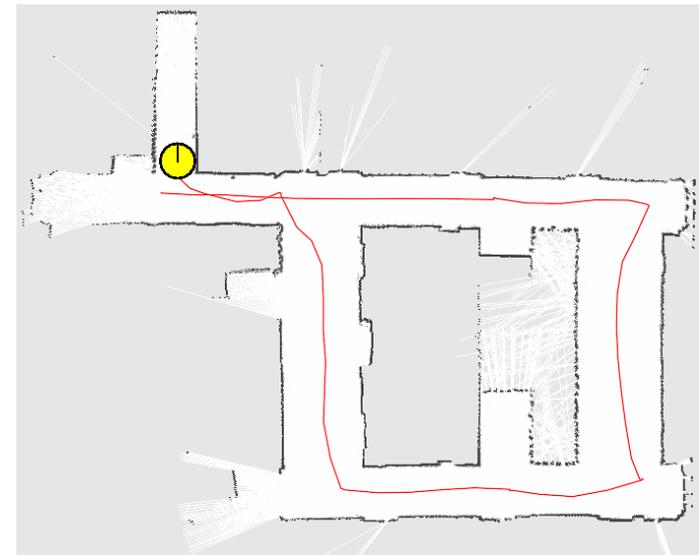
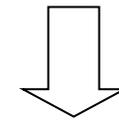
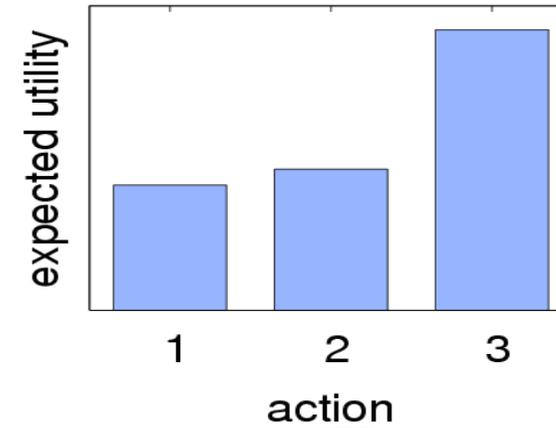
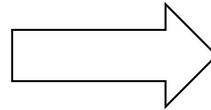
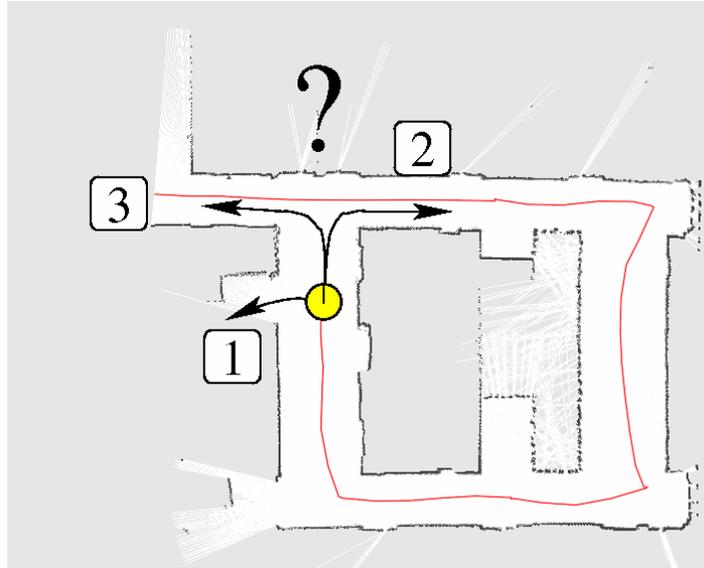
# Dual Representation for Loop Detection

- **Trajectory graph** stores the **path traversed by the robot.**
- **Grid map** represents the **space covered by the sensors.**
- **Loops** correspond to **long paths in the trajectory graph** and **short paths in the geometric map.**

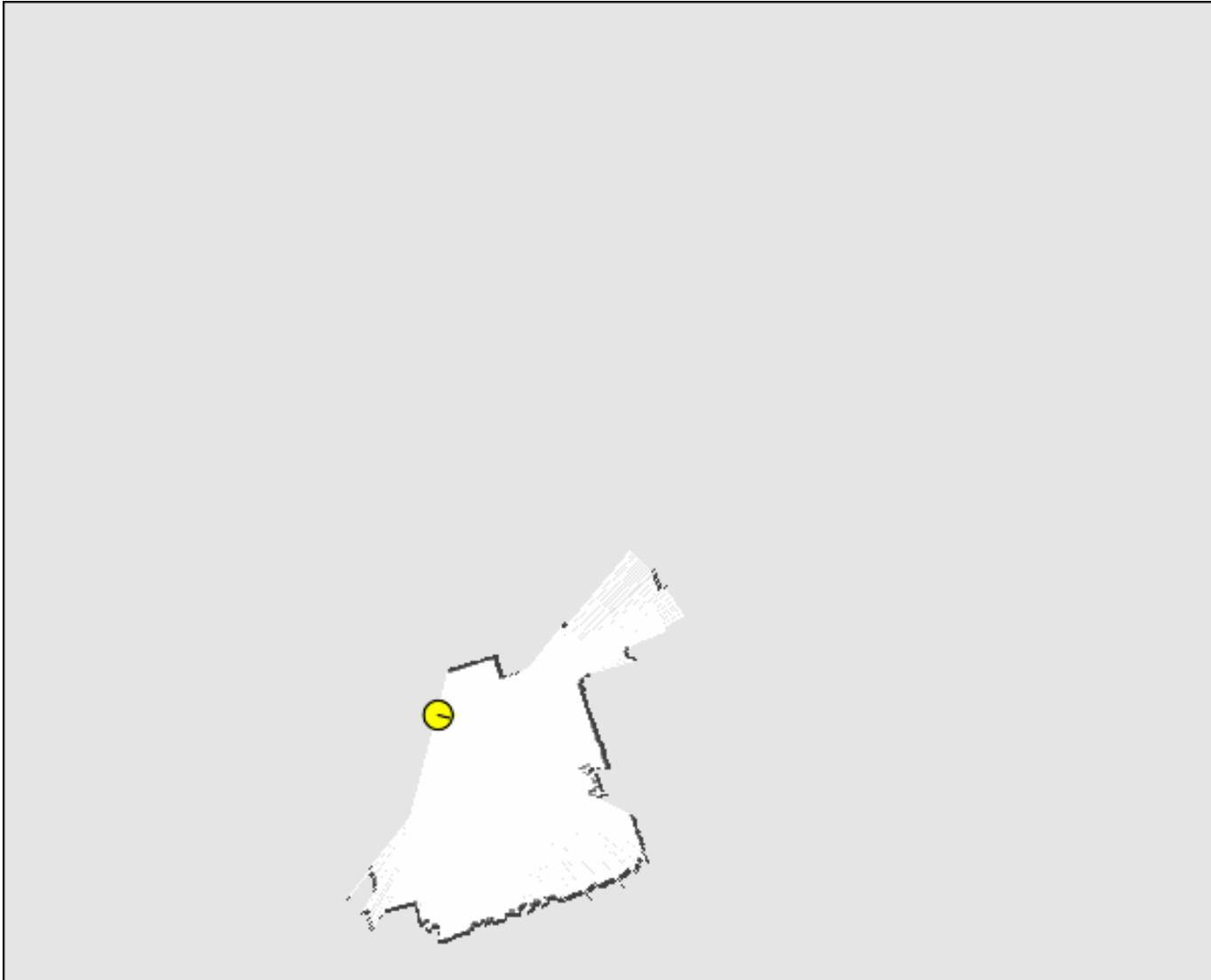
# Dual Representation for Loop Detection



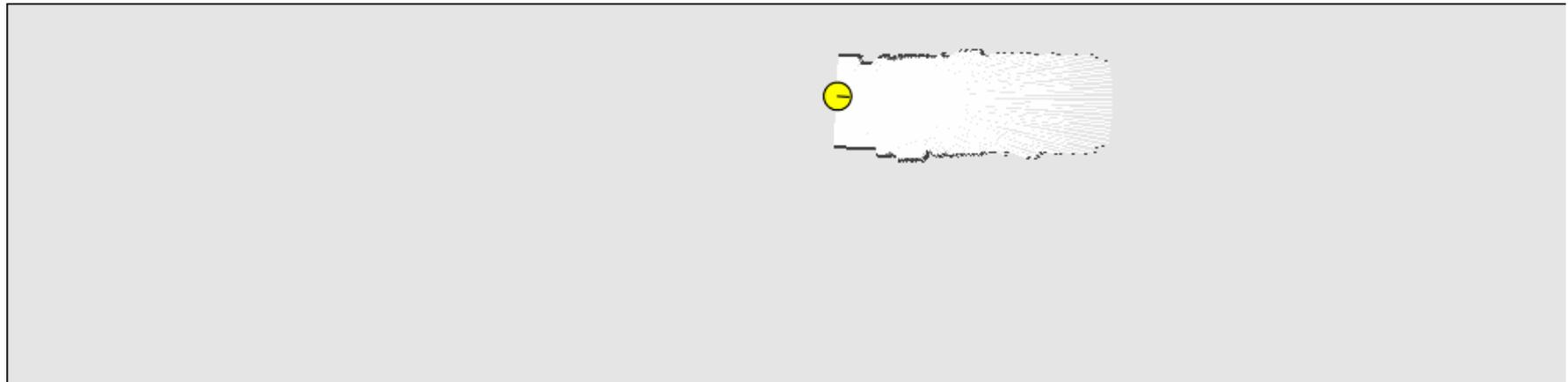
# Application Example



# Real Exploration Example



# Considering Pose Uncertainty during Exploration



# Research Topics

- Large-scale maps
- Resource-constrained systems
- Data association
- Dynamic environments
- Continuous SLAM
- Semantics
- Efficiency of integrated problem
- ...

# Summary

- Probabilistic methods are a powerful tool for realizing robust autonomous systems.
- By reasoning about controls, the given algorithms can get even more effective.
- Probabilistic approaches are highly relevant for building robust navigation systems